UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

# Bert and Distil-Bert Fine Tuning

Student name: *Theodoros Diamantopoulos*
*sdi: sdi1800266*

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

## Contents

## 1. Introduction

This project investigates the use of transformer-based language models for the task of sentiment analysis on English-language tweets. The focus is on two widely used models: BERT and its lightweight counterpart, DistilBERT. Both are part of the Hugging Face Transformers library and have shown strong performance in a variety of natural language processing tasks.

BERT (Bidirectional Encoder Representations from Transformers) is known for its ability to understand context by processing text in both directions. Unlike traditional word embeddings like GloVe, which give each word a fixed vector, BERT produces dynamic embeddings that depend on the surrounding words in a sentence. In this project, BERT is fine-tuned on a labeled dataset of tweets to perform binary sentiment classification.

Alongside BERT, DistilBERT is also explored. DistilBERT is a smaller and faster version of BERT that has been designed to retain most of BERT's performance while reducing the number of parameters and training time. Since many real-world applications have resource constraints, comparing these two models offers insight into the trade-offs between speed and accuracy.

The training process for both models follows a similar structure. Several experiments are run to tune hyperparameters such as learning rate, batch size, and number of training epochs. The goal is to evaluate how these choices impact model performance and to identify which model provides the best balance between efficiency and predictive power. Performance is measured using standard metrics including accuracy, F1-score, and loss over time.

## 2. Data processing and analysis

### 2.1. Pre-processing

Pre-processing is a critical step in any machine learning and AI application as it prepares raw data for model training and ensures that it is clean, structured, and ready for analysis. In this project, various data cleaning techniques were implemented to process the text data and enhance the model's performance.

The following methods were applied during the pre-processing phase:

- **HTML Entity Decoding**: The text was decoded to remove HTML entities. This step ensures that any encoded HTML symbols, such as `&amp;` for & are converted back to their original characters.

- **Lowercasing**: The entire text was converted to lowercase. This simplifies the data by ensuring that words like `Apple` and `apple` are treated as the same token, which helps in reducing redundancy and improving consistency.

- **Email Address Removal**: All email addresses were replaced with a generic placeholder, `xxx@email.com`. This was done to anonymize any personal information and avoid any potential bias that might arise from email addresses.

- **Twitter Usernames Removal**: Usernames (e.g., `@username`) were removed from the text. This is particularly important in tweet datasets where usernames don't contribute to the meaning or sentiment of the message.

- **URL Removal**: Any URLs in the text were replaced with a placeholder (`httpxxx`). Since URLs don't contribute to the sentiment analysis, removing them ensures that they do not affect the model's performance.

- **Punctuation Removal**: All punctuation marks, such as periods, commas, and question marks, were removed. This step ensures that punctuation doesn't interfere with tokenization and word analysis, as it generally doesn't carry meaningful information in this context.

- **Number Removal**: Numbers were also removed from the text. In most natural language processing (NLP) tasks, numerical values are irrelevant unless the task specifically requires them.

- **Non-Alphanumeric Characters and Emoji Removal**: Any non-alphanumeric characters, including emojis, were eliminated from the text to reduce noise in the data. This is essential in datasets containing a mix of standard text and special characters.

- **Whitespace Normalization**: Extra spaces and tabs were removed, ensuring that the text is clean and properly formatted. This step improves the accuracy of tokenization and ensures consistency in text representation.

- **Repetitive Character Removal**: Any characters that appeared three or more times consecutively (e.g., `loooool`) were reduced to just two occurrences (e.g., `lol`). This helps in normalizing exaggerated expressions often seen in informal text.

- **Tokenization**: The text was tokenized using the `TweetTokenizer` from the NLTK library. This tokenizer is particularly suited for handling social media text, as it preserves important features like hashtags and mentions while splitting the text into individual words or tokens.

- **Stopword Removal**: Common stopwords like `the`, `and`, and `is` were removed from the text. These words are usually insignificant in text classification tasks as they do not contribute meaningful information.

- **Lemmatization**: The text was lemmatized using the `WordNetLemmatizer` from NLTK. This step reduces words to their base or root form (e.g., `running` becomes `run`), which helps to reduce dimensionality and improve the model's performance.

After applying these pre-processing steps, the data was transformed into a cleaned and structured format suitable for training machine learning models. The pre-processed text was then stored in a new column (`preprocessed_text`) for further analysis and modeling.

## 2.2. Vectorization

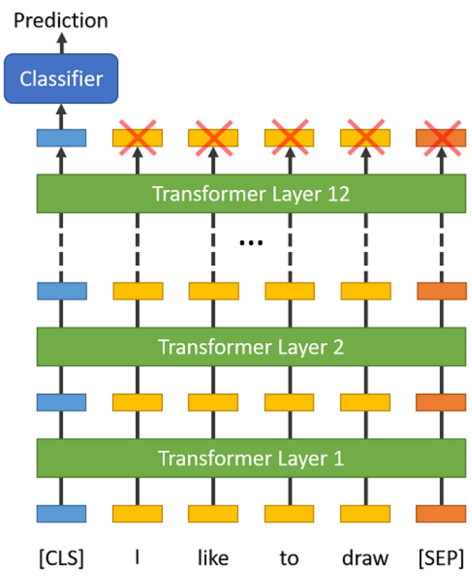
In this project, word vector representations are obtained using contextualized embeddings from transformer-based models. Specifically, the `BertTokenizer` is used to tokenize the text for both BERT and DistilBERT, converting each preprocessed sentence into a sequence of token IDs and corresponding attention masks.

Unlike traditional static word embeddings like GloVe, which assign a fixed vector to each word regardless of context, BERT and DistilBERT generate dynamic, sentence-level embeddings that depend on the surrounding words. These models take entire sentences as input and compute token representations that reflect contextual meaning.

For both models, the tokenizer splits input text into *subword tokens* (e.g., `playing` becomes `play` and `##ing`), adds special tokens such as `[CLS]` and `[SEP]`, and pads or truncates sentences to a fixed maximum length. The resulting token IDs and attention masks are then passed to the transformer encoder.

- In **BERT**, the full transformer architecture (12 layers, 110M parameters) is used to compute contextual embeddings, which are then passed through a classification head for binary sentiment prediction.
- In **DistilBERT**, a smaller version of the transformer (6 layers, 66M parameters) is used. Despite its reduced size, it retains most of BERT's performance while significantly improving speed and memory efficiency.

This process allows both models to produce high-quality, context-aware representations of text that can be fine-tuned on the downstream sentiment classification task.



## 3. Algorithms and Experiments

In this section, a series of experiments are conducted to explore and fine-tune key hyperparameters of transformer-based sentiment classifiers. As in the previous project, experimentation begins with a baseline model and progresses by gradually modifying various training parameters in order to understand their effect on the model's performance.

**The hyperparameters that will be tuned are:**

1. **Training Epochs and Batch Size:** The number of training epochs controls how long the model is trained. Fewer epochs may lead to underfitting, while too many can cause overfitting. The batch size affects training stability and speed, and may influence convergence behavior.
2. **Learning Rate:** This defines the step size at each iteration while moving toward a minimum of the loss function. A well-chosen learning rate is crucial for achieving convergence.
3. **Optimizer Settings:** We primarily use the AdamW optimizer, and experiments include adjusting its epsilon value for numerical stability.
4. **Warm-up Steps:** A small number of initial training steps during which the learning rate increases linearly from zero. This helps prevent sudden large updates to the model early in training, which can destabilize learning.

Each of these parameters is tested independently while keeping the rest constant to evaluate its isolated effect. Metrics such as accuracy, loss, F1-score, and ROC-AUC are used to assess performance.

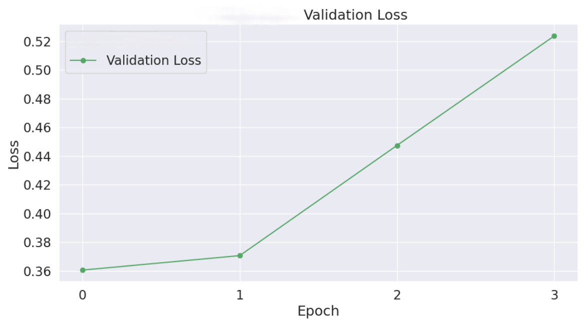
### 3.1. Experiments for Bert

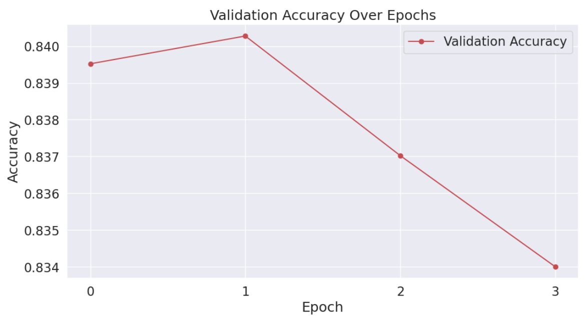
**Link for BERT Model code:** https://www.kaggle.com/code/sdi1800266/sdi1800266-hw3-bert

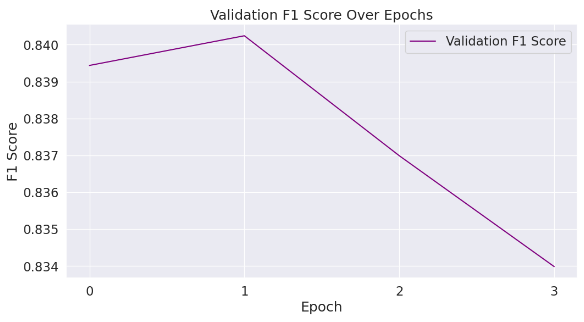
***3.1.1. Baseline Model.*** First, a baseline model was created using the pretrained `bert-base-uncased` model from Hugging Face, fine-tuned on the preprocessed tweet dataset. For this baseline, the standard training loop was implemented with fixed hyperparameters inspired by the course material and example notebooks (`learning rate` = 2e-5, `epochs` = 4, `optimizer` = `AdamW(model.parameters(), lr=2e-5, eps=1e-8)`). A linear learning rate scheduler was used without warm-up steps (`get_linear_schedule_with_warmup(...,` `num_warmup_steps=0)`). This configuration serves as a point of comparison for all future experiments and tuning efforts.

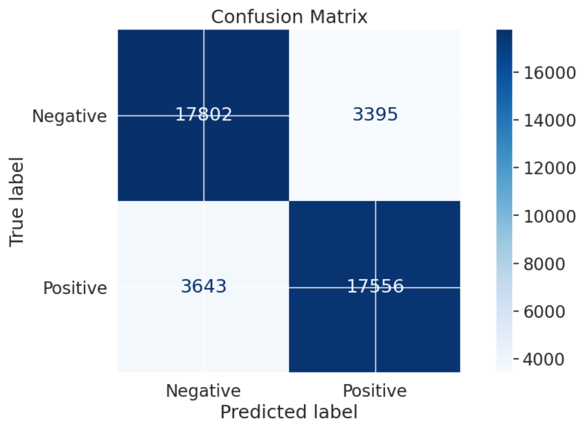
Table 1: Training and Validation Metrics Across Epochs for Baseline Model

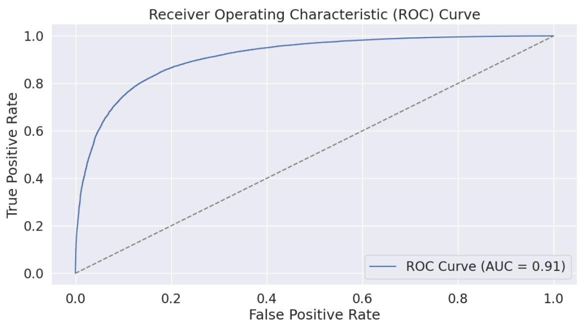
| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.40 | 0:27:10 | 0.36 | 0:02:51 | 0.84 | 0.84 / 0.84 |
| 2 | 0.30 | 0:27:28 | 0.37 | 0:02:51 | 0.84 | 0.84 / 0.84 |
| 3 | 0.21 | 0:27:28 | 0.45 | 0:02:51 | 0.84 | 0.84 / 0.84 |
| 4 | 0.14 | 0:27:28 | 0.52 | 0:02:51 | 0.83 | 0.83 / 0.83 |

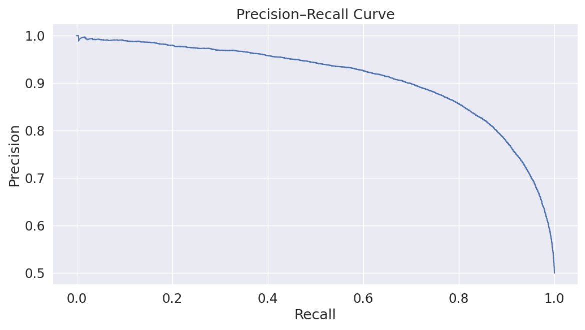












**Loss over epochs:** While the training loss consistently decreases, the validation loss starts increasing after the first epoch, clearly indicating the onset of overfitting. This divergence highlights the need for early stopping or additional regularization techniques to prevent degradation on unseen data. For that reason, epochs will be decreased to 3 for the rest following experiments.

**Accuracy over epochs:** The validation accuracy peaks early and gradually declines over the following epochs. This again reflects overfitting, as the model becomes overly specialized on training data and begins to lose generalization capability.

**F1 score over epochs:** The validation F1 score follows a similar trend to accuracy, rising initially and then deteriorating. This trend confirms that the model's balance between precision and recall diminishes with continued training.

**Confusion matrix:** The confusion matrix indicates fairly symmetric performance across the two sentiment classes, with a moderate number of false positives and false negatives. However, the model shows a slightly better true positive rate compared to true negative, suggesting a minor bias toward the positive class.

**ROC curve:** The ROC curve achieves an AUC of 0.91, indicating strong classification performance. The model distinguishes well between classes, especially at lower false positive rates, and this supports its ability to provide reliable probabilistic outputs.

**Precision–Recall curve:** The precision–recall curve shows excellent precision at high-confidence predictions (low recall), gradually decreasing as the model is forced to retrieve more samples. This is particularly valuable in real-world settings where high precision is preferable, such as spam filtering or review moderation.

***3.1.2. Adding Warmup Steps = 0.1 * total steps.*** In the previous experiment, we observed signs of overfitting, as the validation loss began increasing and both accuracy and F1 score declined after the second epoch. Based on this observation, in this experiment we reduce the number of training **epochs from 4 to 3** to mitigate overfitting. Additionally, we introduce a small warm-up phase by setting `num_warmup_steps` to 10% of the total training steps. All other hyperparameters remain unchanged (*lr=2e-5, eps=1e-8, batch_size=32*).

Table 2: Training and Validation Metrics Across Epochs with Warmup Steps

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.42 | 0:25:42 | 0.36 | 0:02:31 | 0.84 | 0.84 / 0.83 |
| 2 | 0.31 | 0:26:06 | 0.36 | 0:02:28 | 0.84 | 0.84 / 0.84 |
| 3 | 0.22 | 0:25:54 | 0.42 | 0:02:29 | 0.84 | 0.84 / 0.84 |

**Comparison:** In the previous experiment (Table 1), there were clear signs of overfitting, as the validation loss increased steadily and both accuracy and F1 scores slightly declined after the second epoch. The adjustments in Table 2 were made to address this issue. As shown, the validation loss in the updated experiment remains more stable, and both accuracy and F1 scores are consistent across all three epochs. Although the metrics are very similar between the two experiments, the warmup-based approach appears to offer slightly better stability and prevents overfitting in later epochs, making it a more reliable training configuration for this task.

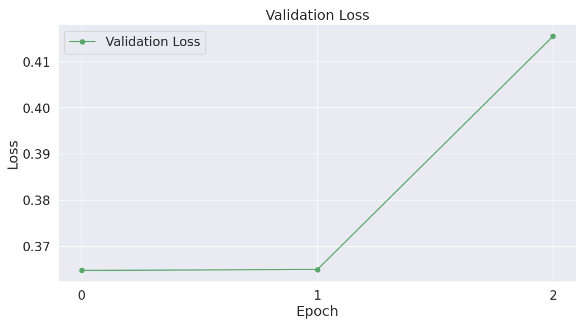


Figure 1: Validation Loss over Epochs (Experiment 2)

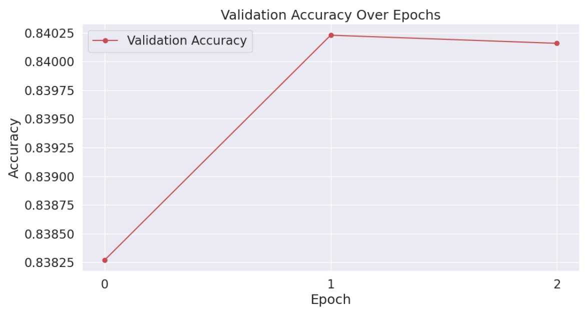


Figure 2: Validation Accuracy over Epochs (Experiment 2)

***3.1.3. Increasing Learning Rate to 3e-5.*** In the previous experiment, a slight improvement in stability was achieved by introducing a warmup phase equal to ten percent of the total training steps. Building on that setup, this experiment increases the learning rate from 2e-5 to 3e-5, while keeping all other hyperparameters unchanged(*eps=1e-8*, *batch_size=32*). The aim is to investigate whether a slightly higher learning rate can lead to faster convergence or improved generalization without causing instability.

Table 3: Training and Validation Metrics Across Epochs Learning Rate Increase(3e - 5)

| **Epoch** | **Train Loss** | **Train Time** | **Val Loss** | **Val Time** | **Accuracy** | **F1 (Neg/Pos)** |
|---|---|---|---|---|---|---|
| 1 | 0.4145 | 0:25:07 | 0.3640 | 0:02:24 | 0.8380 | 0.84 / 0.83 |
| 2 | 0.3000 | 0:25:08 | 0.3692 | 0:02:24 | 0.8407 | 0.84 / 0.84 |
| 3 | 0.1867 | 0:25:07 | 0.4470 | 0:02:25 | 0.8390 | 0.84 / 0.84 |

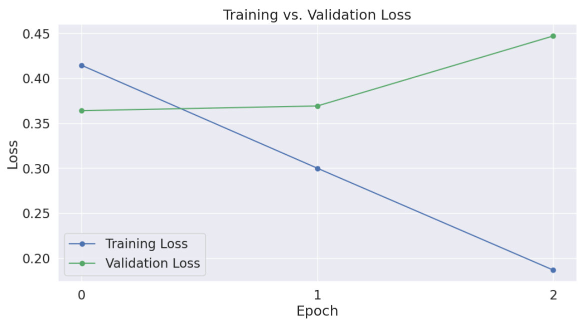


Figure 3: Training and validation loss across epochs for learning rate experiment.

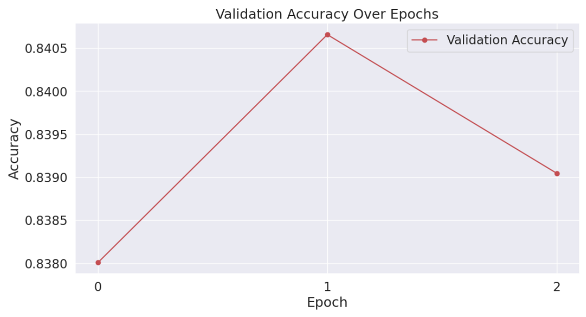


Figure 4: Validation accuracy across epochs for learning rate experiment.

**Comparison:** Compared to the previous experiment with a learning rate of 2e-5, this experiment increases the learning rate to 3e-5 while keeping the warmup phase and all other hyperparameters unchanged. As shown in Figure 3 and Figure 4, the model converges more quickly in terms of training loss. However, the validation loss begins to increase after the first epoch, reaching 0.4470 by the end of training, which indicates early overfitting. Validation accuracy follows a similar trend: it peaks slightly in the second epoch and then drops. Despite the faster training, this setup does not lead to any measurable improvement in validation performance. These results suggest that **increasing the learning rate** may hurt stability without offering real benefits. For this reason, the learning rate will be reset to 2e-5 in subsequent experiments.

***3.1.4. Decreasing Batch Size to 16.*** In this final experiment, the batch size is reduced from 32 to 16 while keeping all other hyperparameters unchanged(*learning_rate=2e-5, epochs=3, epsilon=1e-8, num_warmup_steps=0.1*(total_steps)*). This adjustment aims to explore whether a smaller batch size can improve generalization by introducing more gradient noise and acting as an implicit regularizer. Since previous experiments indicated slight signs of overfitting at higher epochs, lowering the batch size may help stabilize validation performance and improve robustness. The impact of this change is evaluated by comparing validation loss, accuracy, and F1 score against the best-performing configuration so far.

Table 4: Training and Validation Metrics Across Epochs (Batch Size = 16)

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.4160 | 0:30:59 | 0.3683 | 0:02:47 | 0.84 | 0.84 / 0.83 |
| 2 | 0.3029 | 0:31:04 | 0.3643 | 0:02:47 | 0.84 | 0.85 / 0.84 |
| 3 | 0.1945 | 0:31:03 | 0.4384 | 0:02:47 | 0.84 | 0.84 / 0.84 |

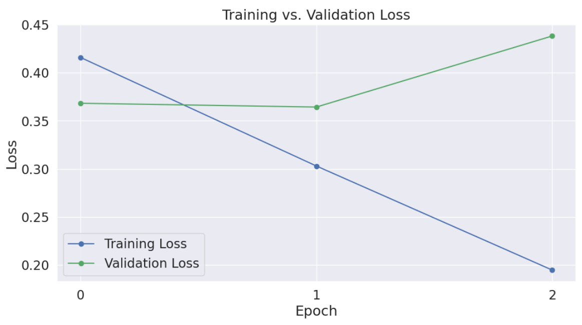


Figure 5: Training and validation loss for experiment with batch size 16.

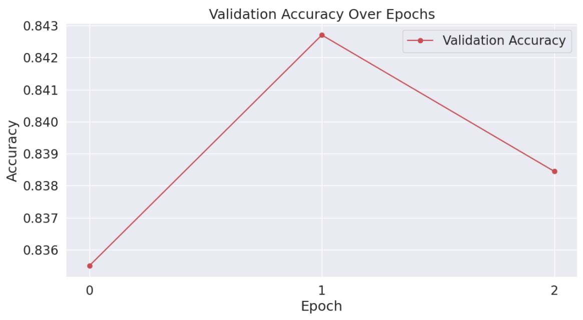


Figure 6: Validation accuracy across epochs for experiment with batch size 16.

**Comparison:** Compared to previous experiments, the training loss decreases more sharply, indicating faster learning per epoch. However, the validation loss increases steadily, especially after the second epoch, which suggests a tendency toward overfitting. The validation accuracy and F1 scores remain nearly identical to the earlier setup with batch size 32. These results suggest that reducing the batch size does not offer significant performance improvements under the current configuration and slightly worsens validation stability. Therefore, the original batch size of 32 is preferred for its balance between efficiency and generalization.

***3.1.5. Conclusion for Bert .*** Throughout this project, multiple experiments were conducted to fine-tune the performance of the BERT model for tweet sentiment classification. Beginning with a baseline configuration, we explored adjustments such as introducing warmup steps, modifying the learning rate, and changing the batch size. The introduction of a warmup phase (10% of total steps) proved beneficial for training stability, while increasing the learning rate to 3e-5 led to faster convergence but did not improve generalization. Reducing the batch size to 16 also offered no significant performance gains and slightly worsened validation stability. After evaluating all setups, the configuration with **learning rate = 2e-5**, **epsilon = 1e-8**, **batch size = 32**, and **warmup steps = 0.1 * total steps** emerged as the most balanced and reliable, consistently achieving strong validation accuracy and F1 scores across epochs without signs of overfitting.

Table 5: Training and Validation Metrics Across Epochs (Optimal Model)

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.4170 | 0:28:44 | 0.3647 | 0:03:01 | 0.84 | 0.84 / 0.83 |
| 2 | 0.3103 | 0:28:48 | 0.3649 | 0:03:02 | 0.84 | 0.84 / 0.84 |
| 3 | 0.2209 | 0:28:49 | 0.4155 | 0:03:02 | 0.84 | 0.84 / 0.84 |

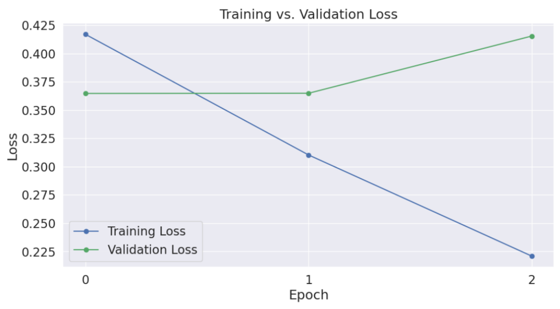


Figure 7: Training and validation loss over epochs (optimal model).

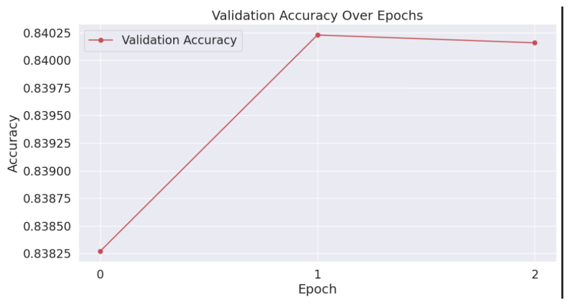


Figure 8: Validation accuracy over epochs (optimal model).

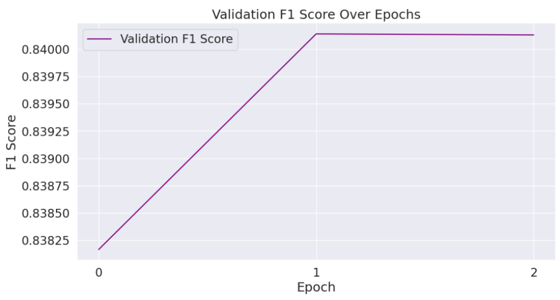


Figure 9: Validation F1 score across epochs (optimal model).

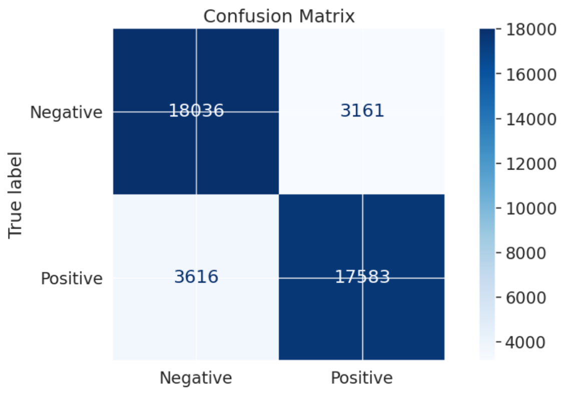


Figure 10: Confusion matrix on validation set (optimal model).

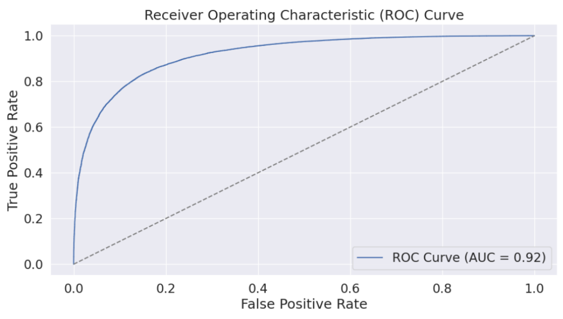


Figure 11: ROC curve with AUC score (optimal model).

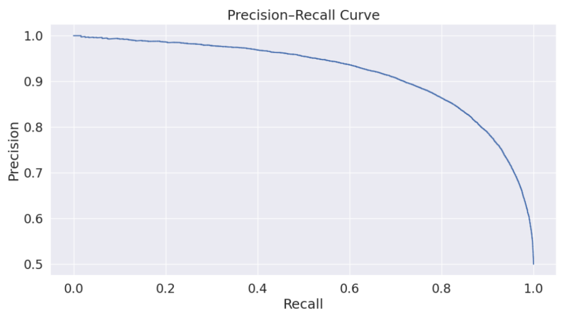


Figure 12: Precision–Recall curve (optimal model).

As seen in the final plots, validation accuracy peaked at 0.8427, and the validation F1 score remained around 0.84 across all epochs. Additionally, the ROC curve returned an AUC of 0.92, and the precision-recall curve maintained high precision even as recall increased, indicating strong discriminative ability. The confusion matrix also shows a balanced classification between positive and negative labels. These results suggest that this configuration offers the best trade-off between convergence speed, generalization, and classification quality, making it the most effective BERT fine-tuning setup identified in this study.

### 3.2. Experiments for DistilBERT

**Link for DistilBERT:** https://www.kaggle.com/code/sdi1800266/sdi1800266-hw3-distilbert

***3.2.1. Baseline Model.*** First, a baseline model was created using the pretrained distilbert-base-uncased model from Hugging Face fine-tuned on the preprocessed tweet dataset. For this baseline, the standard training loop was implemented with fixed hyperparameters in- spired by the course material and example notebooks but also the previous experiments (*learning rate = 2e-5, epochs = 3,optimizer=AdamW, epsilon=1e-8, num_warmup_steps=0.1*(tota*

Table 6: Training and Validation Metrics Across Epochs for DistilBERT Baseline

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.4306 | 0:14:23 | 0.3786 | 0:01:29 | 0.8306 | 0.83 / 0.83 |
| 2 | 0.3318 | 0:14:28 | 0.3733 | 0:01:29 | 0.8342 | 0.83 / 0.83 |
| 3 | 0.2571 | 0:14:27 | 0.4104 | 0:01:29 | 0.8318 | 0.83 / 0.83 |

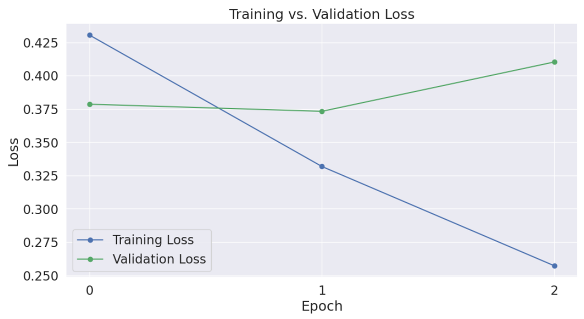


Figure 13: Training vs. Validation Loss (DistilBERT Baseline)

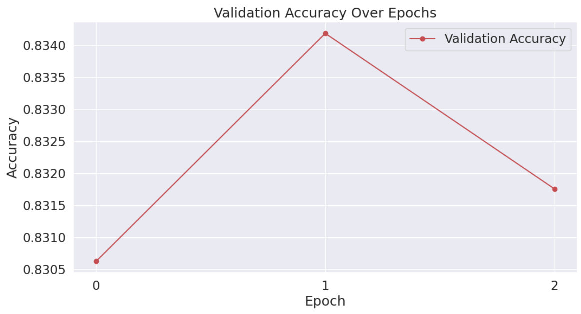


Figure 14: Validation Accuracy Over Epochs (DistilBERT Baseline)

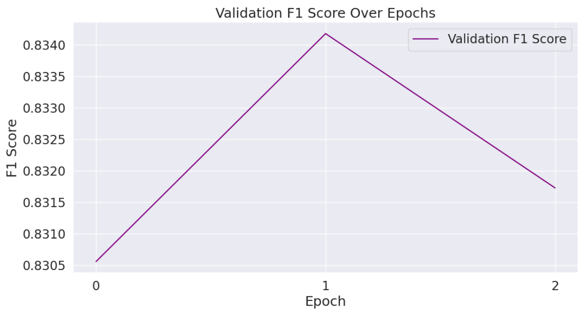


Figure 15: Validation F1 Score Over Epochs (DistilBERT Baseline)

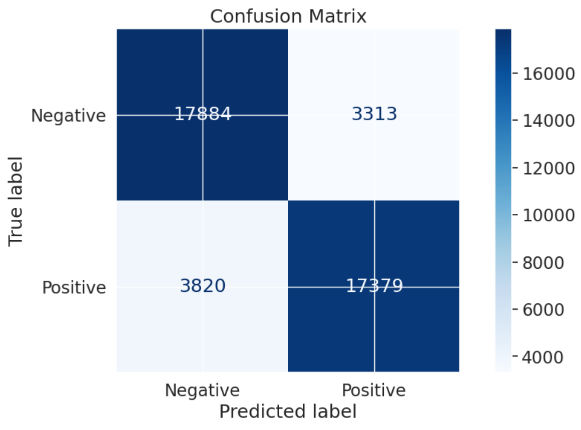


Figure 16: Confusion Matrix on Validation Set (DistilBERT Baseline)

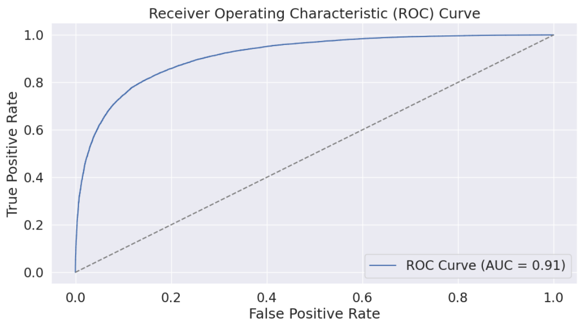


Figure 17: ROC Curve (AUC = 0.91) (DistilBERT Baseline)

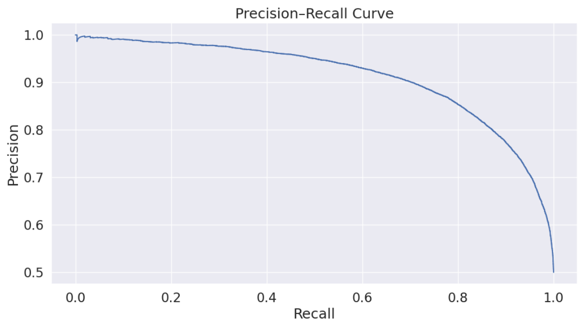


Figure 18: Precision–Recall Curve (DistilBERT Baseline)

**Loss over epochs:** The training loss for the baseline `DistilBERT` model decreases steadily across epochs, from 0.4306 to 0.2571. However, the validation loss slightly increases after the second epoch (from 0.3733 to 0.4104), suggesting the onset of mild overfitting. This behavior indicates that while the model continues to optimize on the training set, its generalization to unseen data slightly deteriorates after the optimal point.

**Accuracy over epochs:** Validation accuracy improves from epoch 1 to epoch 2, peaking at 83.42%, before slightly declining in epoch 3. This pattern suggests that epoch 2 offers the best generalization capability, while continued training may lead to diminishing returns or overfitting effects.

**F1 score over epochs:** The F1 score follows a similar trajectory to accuracy, with both classes (positive and negative) achieving balanced performance throughout. The highest F1 scores are observed at epoch 2, confirming it as the optimal trade-off point between precision and recall.

**Confusion matrix:** The confusion matrix reflects a balanced classification across both sentiment classes, with 17,884 true negatives and 17,379 true positives. The misclassifications are relatively symmetric, indicating that the model does not exhibit a strong bias toward either class.

**ROC curve:** The ROC curve returns an AUC score of 0.91, demonstrating excellent ability to separate the two classes. The model performs particularly well at low false positive rates, making it suitable for high-stakes applications where such errors must be minimized.

**Precision–Recall curve:** The precision–recall curve remains strong across the recall range. Precision is especially high at lower recall levels, gradually declining as the model retrieves more examples. This makes the model a good candidate for applications where high precision is more important than recall, such as sentiment moderation or spam detection.

### 3.2.2. *Learning Rate increase to 5e-5.*

In the previous experiment, the baseline `DistilBERT` model was fine-tuned using a learning rate of 2e-5, which yielded stable and balanced performance. In this experiment, we investigate the impact of a higher learning rate by increasing it to 5e-5, while keeping all other hyperparameters unchanged (*epochs = 3, batch_size = 32, epsilon = 1e-8, warmup steps =* 10% *of total training steps*). The goal is to evaluate whether a more aggressive learning rate can accelerate convergence and enhance model generalization without introducing instability or overfitting.

Table 7: Training and Validation Metrics Across Epochs for DistilBERT (Learning Rate = $5 \times 10^{-5}$)

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.4269 | 0:13:26 | 0.3838 | 0:01:22 | 0.8234 | 0.81 / 0.83 |
| 2 | 0.3154 | 0:13:45 | 0.3786 | 0:01:22 | 0.8336 | 0.83 / 0.83 |
| 3 | 0.1911 | 0:13:44 | 0.4668 | 0:01:22 | 0.8301 | 0.83 / 0.83 |

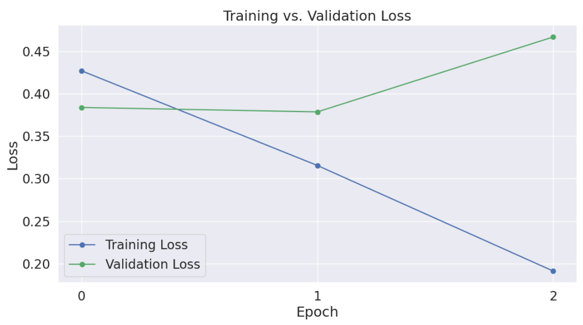


Figure 19: Training vs. Validation Loss (Learning Rate = 5e-5)

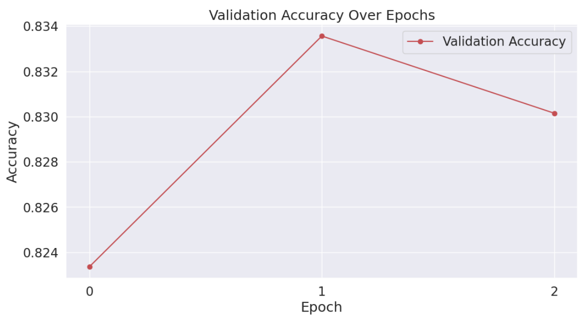


Figure 20: Validation Accuracy Over Epochs (Learning Rate = 5e-5)

**Comparison:** Increasing the learning rate from 2e-5 to 5e-5 led to faster convergence, with a notably lower training loss across epochs. However, the validation loss began to increase after the second epoch, indicating early signs of overfitting. Validation accuracy peaked at epoch 2 before slightly declining, showing that although the higher learning rate improved learning speed, it also introduced instability. This trade-off highlights the importance of careful learning rate selection.

***3.2.3. Decreasing Batch Size to 16.*** Following the findings of the previous experiments, the learning rate will stay fixed at 2e-5 due to its stable and well-generalized performance. In this experiment, the batch size is reduced from 32 to 16, while all other hyperparameters remain unchanged (*epochs = 3, epsilon = 1e-8, warmup steps = 10% of total training steps*). The objective is to evaluate whether a smaller batch size can improve generalization by introducing additional training noise, which may act as a form of regularization.

Table 8: Training and Validation Metrics Across Epochs for DistilBERT with Batch Size 16

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.4270 | 0:16:34 | 0.3774 | 0:01:29 | 0.8302 | 0.83 / 0.83 |
| 2 | 0.3245 | 0:16:37 | 0.3732 | 0:01:29 | 0.8361 | 0.84 / 0.84 |
| 3 | 0.2354 | 0:16:36 | 0.4275 | 0:01:29 | 0.8320 | 0.83 / 0.83 |

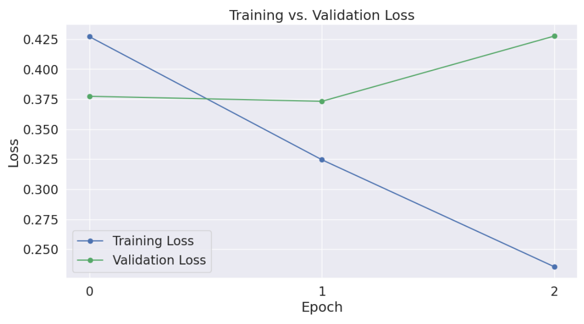


Figure 21: Training vs. Validation Loss (Batch Size = 16)

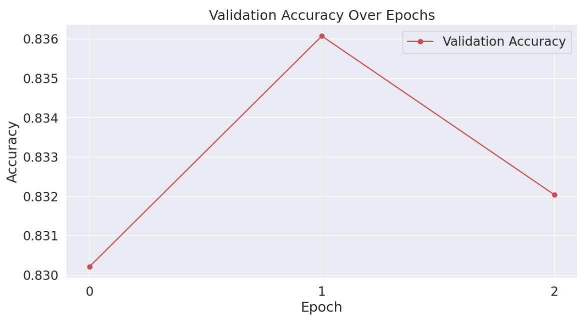


Figure 22: Validation Accuracy Over Epochs (Batch Size = 16)

**Comparison:** Reducing the batch size from 32 to 16, while keeping the learning rate at 2e-5 and all other hyperparameters constant, led to a slight improvement in validation accuracy (from 83.36% to 83.61%) and macro F1-score (from 0.834 to 0.837). The validation loss also decreased marginally from 0.3786 to 0.3732, indicating slightly better generalization. However, the improvement comes at the cost of nearly doubling the number of training steps per epoch (9,275 vs. 4,638). These findings suggest that while smaller batch sizes can offer modest gains in performance, they may require more training time and resources.

***3.2.4. Conclusion for DistilBert.*** After analyzing all experiments, the configuration with learning rate = 2e-5, epsilon = 1e-8, batch size = 16, and warmup steps = 0.1 * total steps was selected as the optimal DistilBERT setup. This model achieved the best trade-off between performance and generalization, while also benefiting from faster training time and lower memory requirements compared to full BERT. The selected configuration provides a highly efficient and effective model, well-suited for real-world applications with limited computational resources.

Table 9: Training and Validation Metrics Across Epochs for DistilBERT with Batch Size 16

| Epoch | Train Loss | Train Time | Val Loss | Val Time | Accuracy | F1 (Neg/Pos) |
|---|---|---|---|---|---|---|
| 1 | 0.4270 | 0:16:34 | 0.3774 | 0:01:29 | 0.8302 | 0.83 / 0.83 |
| 2 | 0.3245 | 0:16:37 | 0.3732 | 0:01:29 | 0.8361 | 0.84 / 0.84 |
| 3 | 0.2354 | 0:16:36 | 0.4275 | 0:01:29 | 0.8320 | 0.83 / 0.83 |

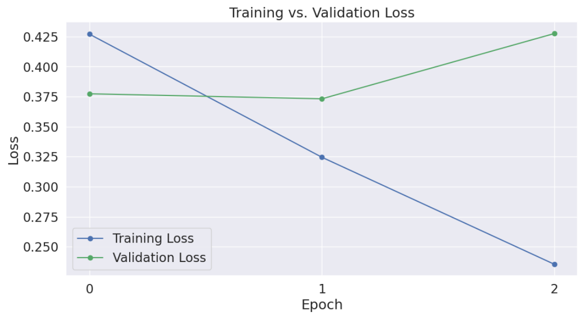


Figure 23: DistilBERT Optimal Model

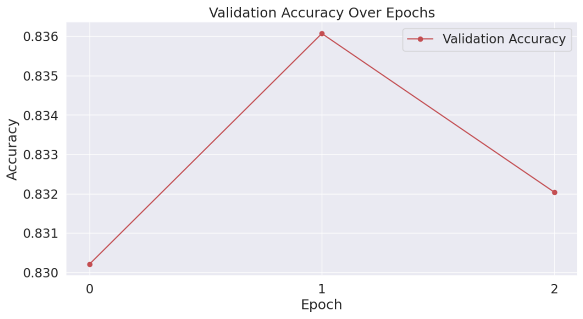


Figure 24: DistilBERT Optimal Model

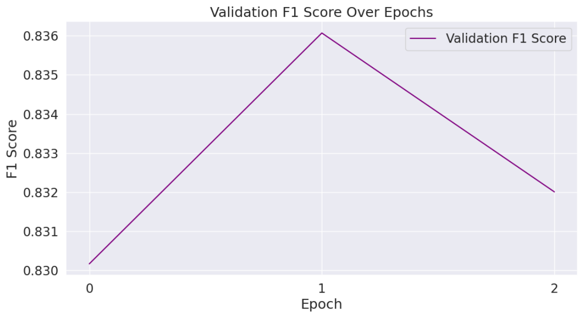


Figure 25: DistilBERT Optimal Model

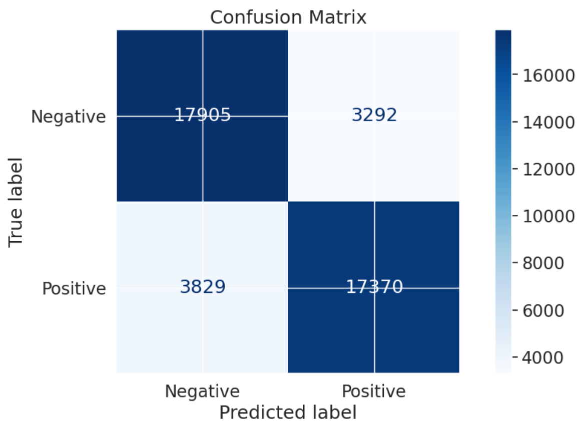


Figure 26: DistilBERT Optimal Model

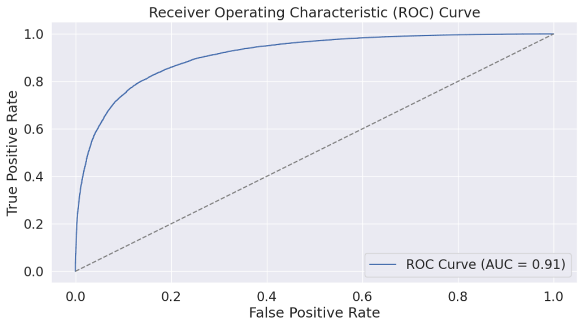


Figure 27: DistilBERT Optimal Model

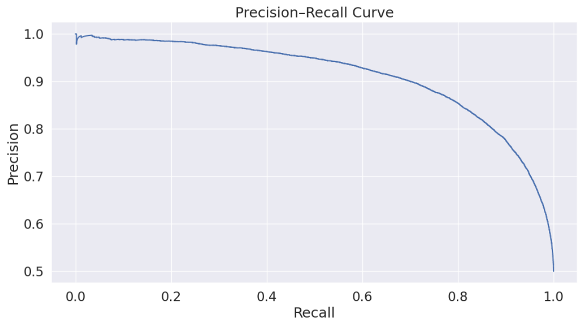


Figure 28: DistilBERT Optimal Model

As shown in the corresponding plots, validation accuracy peaked at 0.8361 and the F1 score reached a maximum of 0.84, indicating strong and stable generalization. The ROC curve yielded an AUC of 0.91, confirming robust class separation, while the precision–recall curve demonstrated sustained precision over a wide recall range. The confusion matrix revealed balanced performance across classes, with minimal misclassification bias. These outcomes demonstrate that this configuration provides the most effective trade-off between convergence, generalization, and classification performance, making it the optimal fine-tuning strategy for DistilBERT in this study.

## 4. Results and Overall Analysis

### 4.1. Comparison: BERT vs DistilBERT

Both **BERT** and **DistilBERT** demonstrated strong performance in the sentiment classification task, with similar validation accuracies and F1 scores across experiments.

- The optimal **BERT** configuration achieved a peak validation accuracy of **0.8427** and an F1 score of **0.84**, along with an **AUC of 0.92**.
- The optimal **DistilBERT** setup reached a slightly lower validation accuracy of **0.8361**, matched the **F1 score of 0.84**, and achieved an **AUC of 0.91**.

Despite the marginal performance gap, **DistilBERT** completed training in nearly half the time and consumed fewer computational resources due to its smaller architecture. These results highlight that **DistilBERT** offers a compelling alternative when computational efficiency is a priority, achieving near-BERT performance at a significantly reduced training cost. However, since this report prioritizes *classification performance over training time*, the **BERT model is considered the best overall configuration**.

### 4.2. Best Trial

Table 10: Performance of the Optimal **BERT** Model Across Epochs

| Epoch | Train Loss | Val Loss | Val Accuracy | Val F1 Score | Duration |
|---|---|---|---|---|---|
| 1 | 0.4170 | 0.3647 | 0.8383 | 0.835 | 26:01 |
| 2 | 0.3103 | 0.3649 | 0.8402 | 0.838 | 26:02 |
| 3 | 0.2209 | 0.4155 | 0.8402 | 0.838 | 26:01 |

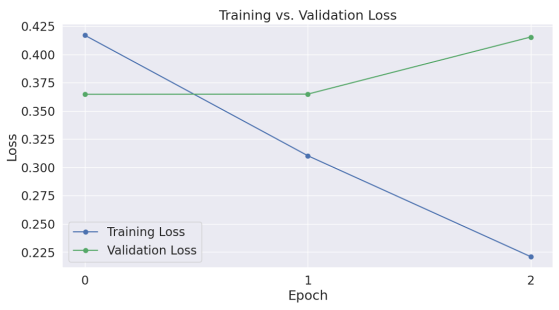


Figure 29: Training and validation loss over epochs (optimal model).

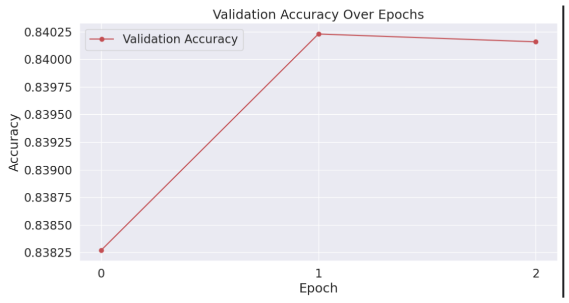


Figure 30: Validation accuracy over epochs (optimal model).

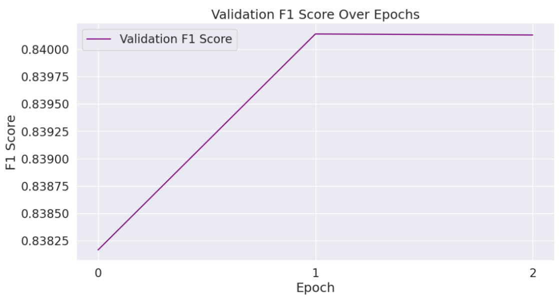


Figure 31: Validation F1 score across epochs (optimal model).

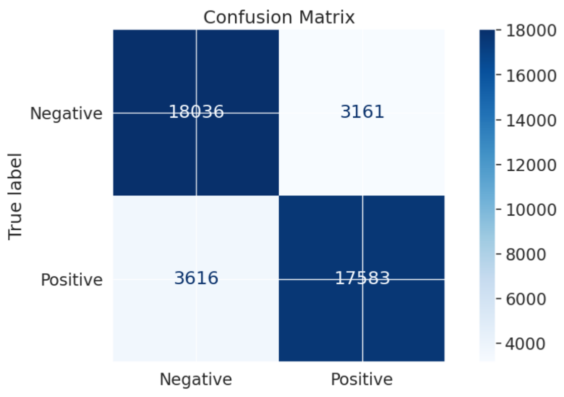


Figure 32: Confusion matrix on validation set (optimal model).

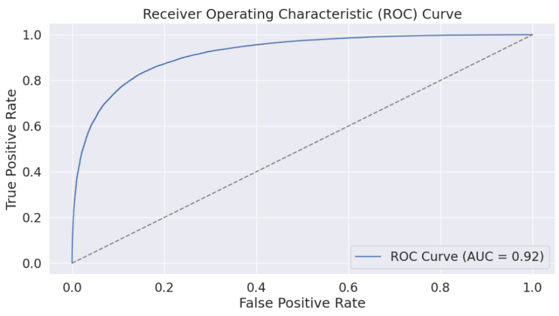


Figure 33: ROC curve with AUC score (optimal model).

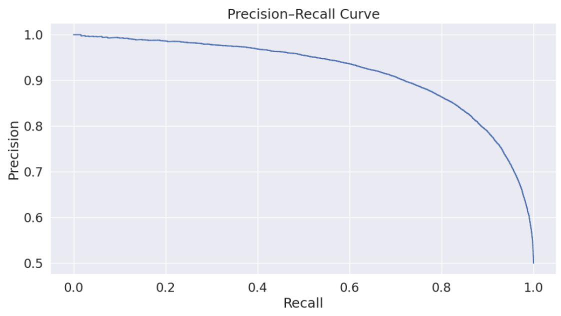


Figure 34: Precision–Recall curve (optimal model).

### 4.3. Comparison with the First Project

The first project implemented a traditional machine learning pipeline using **TF-IDF** features combined with **Logistic Regression**. After several experiments, the best configuration involved bigram tokenization and a large vocabulary size (`max_features = 100,000`), yielding a validation **accuracy and F1 score of approximately 0.80**. While this setup performed reasonably well, it relied heavily on manual feature engineering and lacked the ability to model contextual meaning.

In contrast, the current project fine-tuned **BERT**, a transformer-based model that leverages contextualized word embeddings. After extensive experimentation with hyperparameters such as learning rate, batch size, and warm-up scheduling, the optimal configuration achieved a validation **accuracy and F1 score of 0.84** and an **AUC of 0.92**, demonstrating strong class separability.

Overall, **BERT** outperformed the TF-IDF baseline in every key metric. Although it required more training time and computational resources, the improvement in generalization, robustness, and semantic understanding made BERT the clearly superior approach for this task.

### 4.4. Comparison with the Second Project

The second project utilized a deep learning approach with **pre-trained GloVe embeddings** and a **feedforward neural network** consisting of three hidden layers. After tuning hyperparameters—such as a 300-dimensional embedding size, learning rate of $5 \times 10^{-4}$, and dropout—the model achieved a peak validation **F1 score of 0.81** and a **ROC AUC of 0.88**.

The current project improved upon this by replacing static GloVe vectors with contextualized embeddings from **BERT**, a model pre-trained on large-scale language corpora. Fine-tuning BERT over three epochs with a learning rate of $2 \times 10^{-5}$ and warm-up steps led to superior results, reaching a validation **F1 score of 0.84** and an **AUC of 0.92**.

While the GloVe model was faster to train and computationally lighter, **BERT demonstrated stronger classification performance, particularly in terms of generalization and handling complex language**. These results underscore the effectiveness of transformer-based models for sentiment analysis tasks.

## 5. Bibliography

## References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Geeks for Geeks. Precision-recall (pr) curve in machine learning. https://www.geeksforgeeks.org/precision-recall-curve-ml/, 2025.

[3] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[2][H] [1] [3]