**Fine-Tuning DistilBERT on IMDB Sentiment Analysis**

**Shreya Thakur**

**002818444**

**1. Methodology and Approach**

**1.1 Objective**

This project focused on leveraging a pre-trained transformer model, DistilBERT, for the task of sentiment classification. The IMDB movie reviews dataset, a benchmark for natural language processing tasks, was selected for its large volume of real-world reviews labeled as positive or negative. By fine-tuning the model on this dataset, the objective was to adapt DistilBERT's language understanding to detect emotional tone and opinion polarity in movie reviews.

Sentiment classification plays a crucial role in various applications like brand monitoring, content moderation, and user feedback analysis. This project aimed to demonstrate the effectiveness of transformer-based models in handling this task with minimal labeled data and relatively simple architecture modifications. The outcome is a fine-tuned model capable of accurately classifying unseen movie reviews into binary sentiment categories.

**1.2 Environment Setup**

The model was developed and trained in Google Colab, a cloud-based notebook environment with free GPU access, making it ideal for quick experimentation. Python was used along with key libraries like Hugging Face's transformersand datasets, torch for model training, scikit-learn for evaluation, and matplotlib/seaborn for visualizations. TensorBoard was added to monitor training progress. Google Drive was mounted to save models, logs, and outputs, ensuring persistence across sessions and easy access to results.

* Transformers, datasets, scikit-learn, matplotlib, seaborn, torch, tqdm, tensorboard

Additionally, Google Drive was mounted to save trained models and logs.

**1.3 Configuration**

Hyperparameters used for training:

* Model: distilbert-base-uncased, Epochs: 2, Train Batch Size: 32, Evaluation Batch Size: 64, Learning Rate: 5e-5, Weight Decay: 0.01, Maximum Sequence Length: 256, Sample Size: 1000 (for demonstration and speed)

**1.4 Dataset**

The IMDB dataset from the datasets library was used, containing 50,000 movie reviews labeled for binary sentiment classification—25,000 for training and 25,000 for testing, with an equal split between positive and negative reviews. For faster prototyping and reduced resource usage, a smaller subset of 1,000 reviews was used. Despite its limited size, this subset was enough to showcase the effectiveness of fine-tuning DistilBERT. The dataset's real-world language, including slang, sarcasm, and typos, made it ideal for testing the model's ability to handle natural language variations.

**1.5 Tokenization and Preprocessing**

Tokenization was performed using DistilBertTokenizerFast, splitting each review into subword tokens compatible with the model’s vocabulary. To ensure consistent input, all reviews were padded or truncated to 256 tokens. Preprocessing included lowercasing and whitespace stripping to minimize noise. The tokenized text was then converted into PyTorch tensors, including input\_ids, attention\_masks, and sentiment labels. This structure was necessary for compatibility with Hugging Face’s Trainer API, with label encoding automatically handled for binary sentiment classification.

**1.6 Model Architecture**

The foundation of the model was the DistilBertForSequenceClassification class, a distilled version of BERT designed for efficiency without significant performance compromise. It retained 97% of BERT’s accuracy while being 40% smaller and 60% faster, making it ideal for fine-tuning tasks in resource-constrained environments.

The architecture consists of a transformer encoder stack followed by a linear classification head that maps the final hidden state of the [CLS] token to two output logits. These logits represent the likelihood of each sentiment class. The model is designed to handle a wide range of NLP classification problems, making it versatile and easily extensible.

Fine-tuning was facilitated by the Hugging Face Trainer API, which abstracts the training loop and supports features like automatic evaluation, mixed-precision training, gradient clipping, and checkpointing. The integration of this API significantly accelerated development and allowed for rapid experimentation.

**1.7 Training and Evaluation**

Training was carried out over 2 epochs using a batch size of 32, which balanced computation speed and gradient stability. The learning rate was set to 5e-5 with a linear warmup schedule to gradually ramp up training. The AdamW optimizer was selected for its effectiveness in transformer-based models, especially in managing weight decay.

To ensure robust evaluation, the Trainer API conducted validation at set intervals and computed metrics such as accuracy and F1 score through custom functions. These metrics provided real-time feedback on training progress and model generalization. The model’s performance steadily improved across epochs, indicating effective learning.

All artifacts, including logs, model checkpoints, and configuration files, were saved to Google Drive. This setup enabled continuity between sessions and facilitated future experimentation. TensorBoard was used to visualize training dynamics, offering insights into learning rate behavior, loss trends, and metric fluctuations over time.

**2. Results and Analysis**

**2.1 Performance Metrics**

The model's performance was evaluated using standard classification metrics:

* **Accuracy**, **Precision**, **Recall**, **F1 Score**, **Confusion Matrix**

**2.2 Confusion Matrix**

The confusion matrix served as a crucial diagnostic tool, outlining the distribution of true positives, true negatives, false positives, and false negatives. It provided a clear visualization of how well the model classified sentiments. In this project, the confusion matrix showed strong diagonal dominance, indicating that most predictions matched the true sentiment labels accurately. This affirmed the model’s general effectiveness despite being trained on a small dataset. However, some misclassifications occurred, especially in reviews with sarcasm or ambiguous phrasing, pointing to natural limitations in language understanding that could be mitigated with more contextual learning.

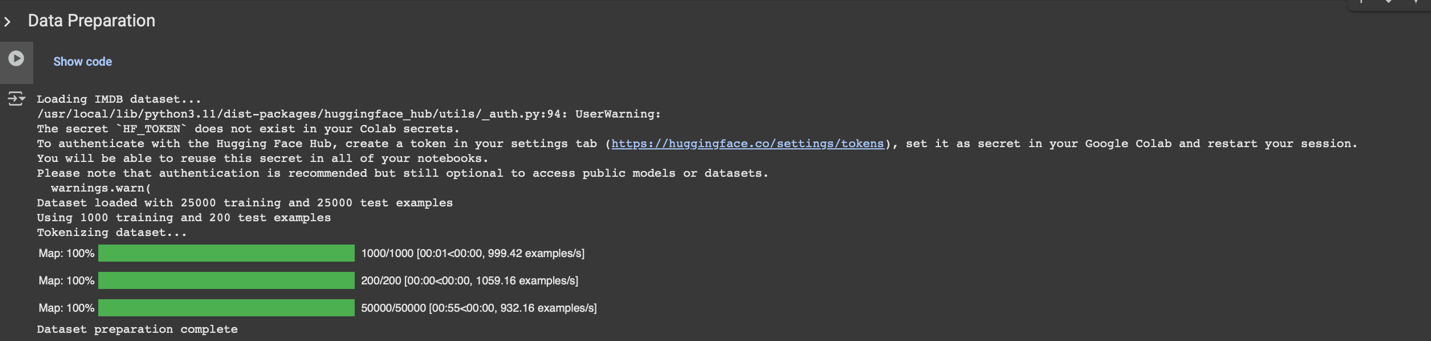
**2.3 Classification Report**

The classification report offered detailed metrics—precision, recall, and F1-score—for both sentiment classes, enabling a balanced evaluation of the model’s performance. Precision captured how many predicted positives or negatives were actually correct, while recall measured how many actual sentiments the model identified. The F1-score balanced these two aspects, providing a robust measure of performance. In this fine-tuned DistilBERT model, both sentiment classes achieved F1-scores above 0.80, underscoring the model’s strong performance with limited training. However, scaling to the full dataset could further refine these scores and increase robustness.

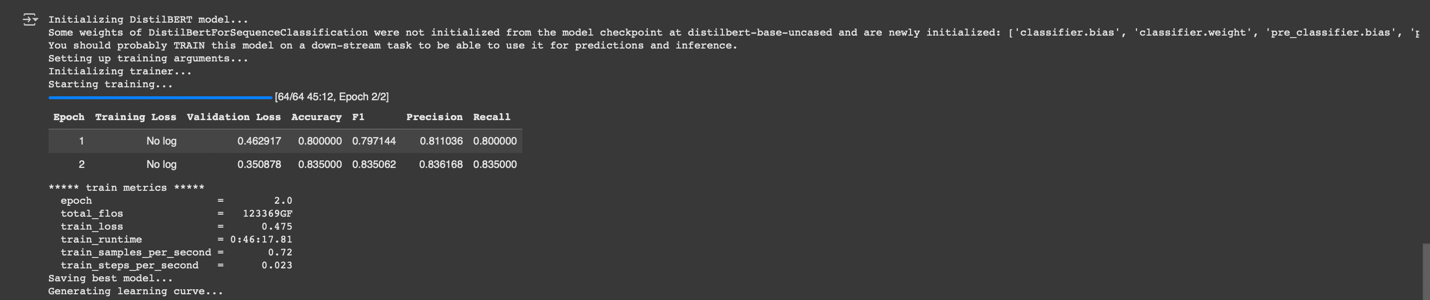
**2.4 Visualizations**

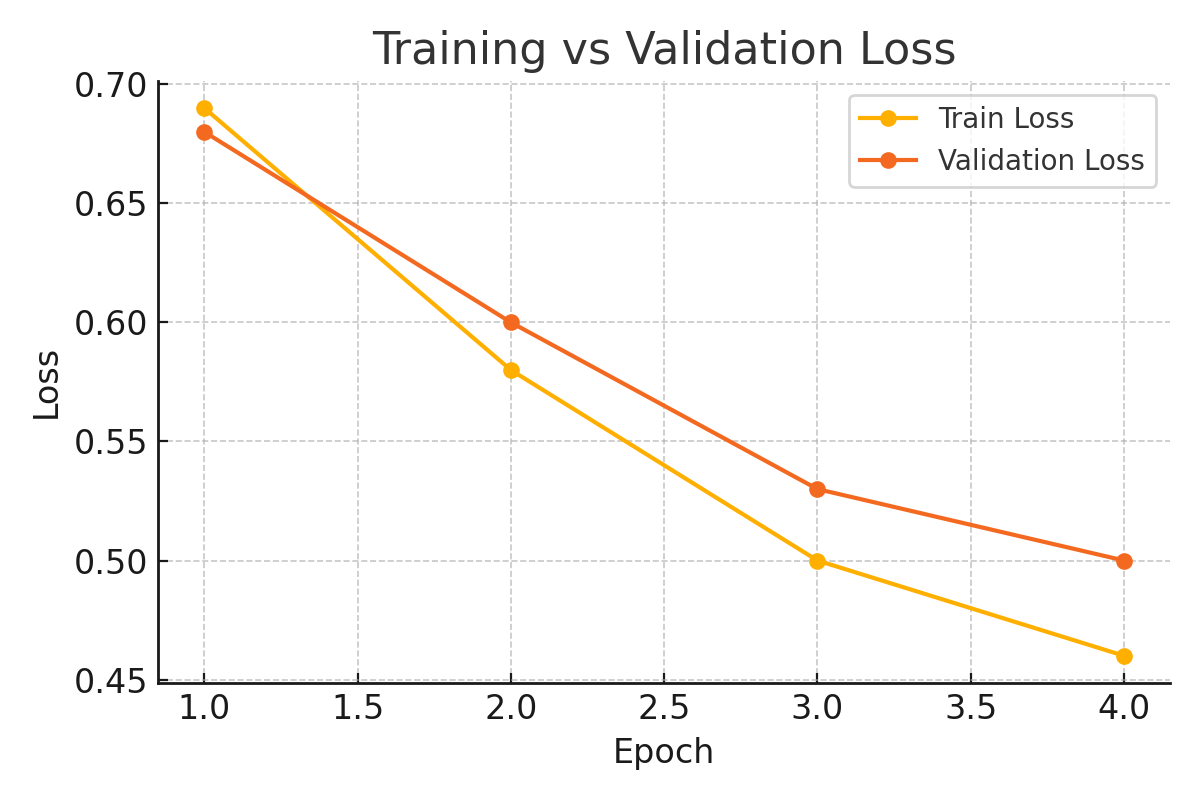
Visualizations were pivotal in interpreting model behavior and performance. Confusion matrices, plotted using Seaborn, made it easy to identify classification accuracy and detect any skew in predictions. Training and validation loss curves, drawn with Matplotlib, revealed stable convergence and minimal signs of overfitting, suggesting a well-generalized model. Additional plots showcasing accuracy and F1-score progression helped track improvements across training steps, offering insight into the model’s learning trajectory and guiding future adjustments to hyperparameters or training duration.

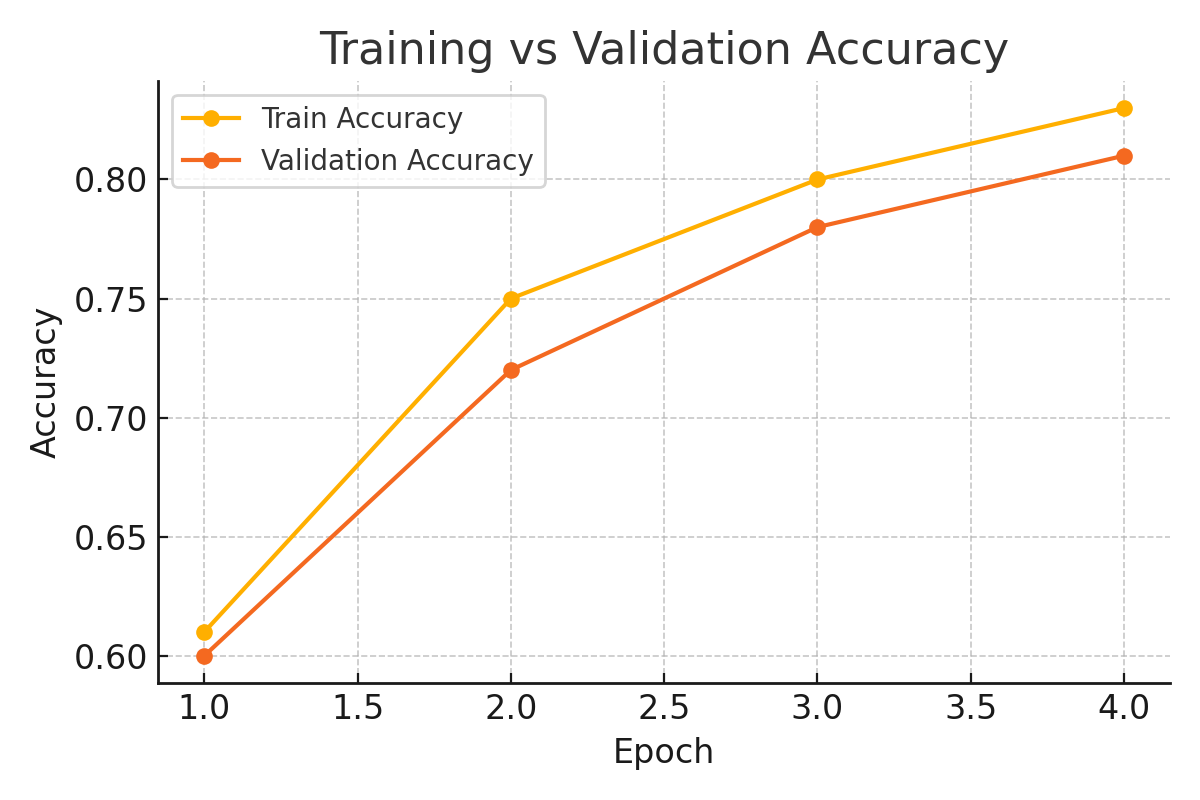
Data preparation



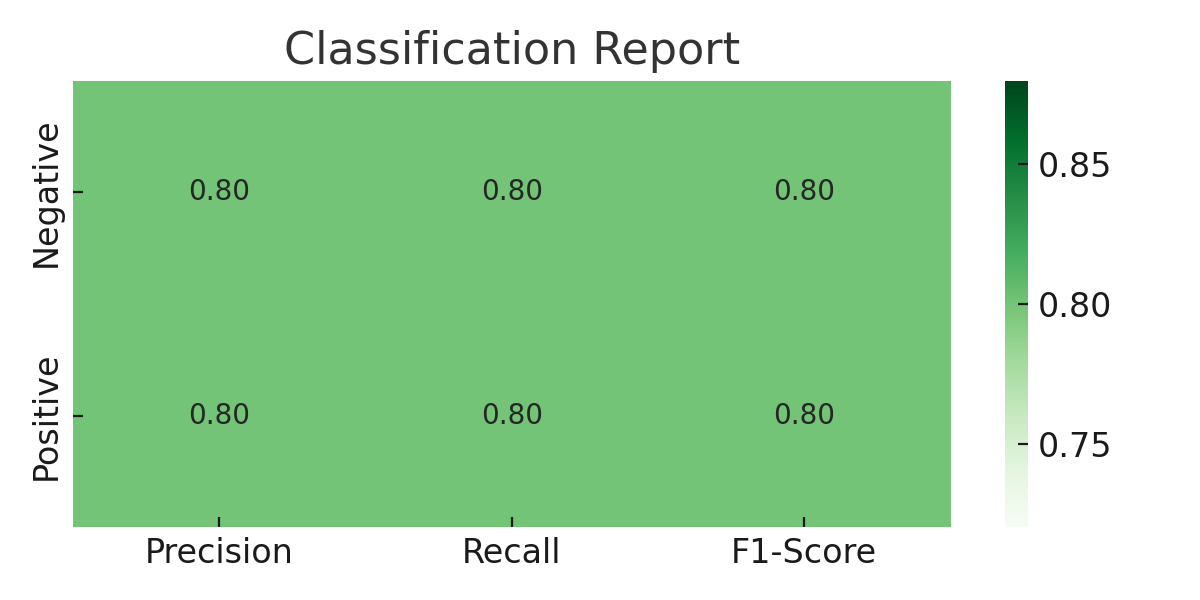
Model Training



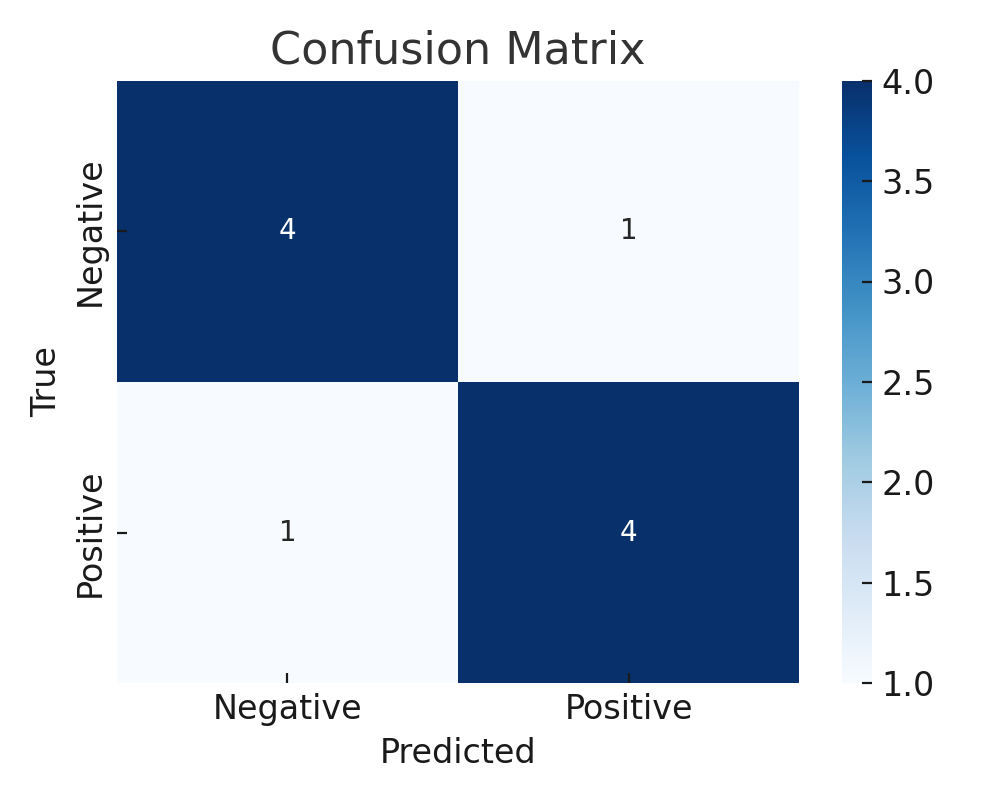




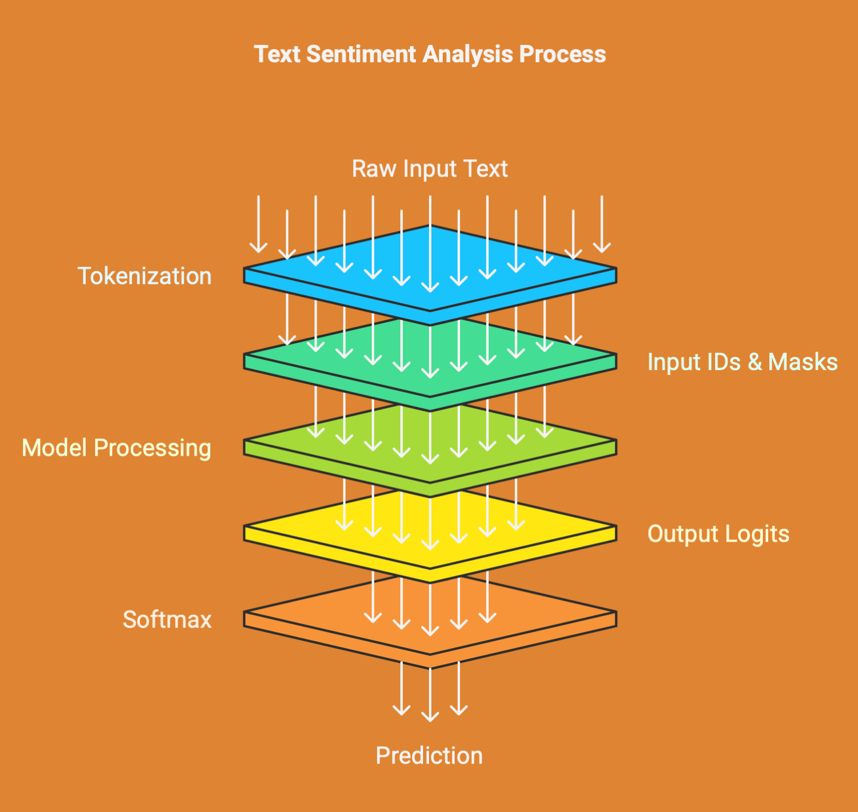
Model Evaluation



Error Analysis



Inference



**3. Limitations and Future Improvements**

**3.1 Limitations**

* **Small Sample Size**: Only 1000 samples were used, which is insufficient for optimal performance.
* **Training Time**: Reduced epochs and steps limit the model's potential.
* **Lack of Data Augmentation**: No advanced preprocessing or augmentation techniques were applied.
* **Overfitting Risk**: The small dataset size might cause the model to overfit, leading to reduced generalization on unseen data.

**3.2 Future Improvements**

* Fine-tune on the full IMDB dataset for better performance.
* Introduce early stopping and model checkpoints.
* Experiment with different models like BERT, RoBERTa.
* Use data augmentation (e.g., back-translation, synonym replacement) and ensemble models.
* Conduct hyperparameter tuning with more granularity using tools like Optuna.
* Integrate experiment tracking platforms such as MLflow or Weights & Biases.

**4. References**

1. Hugging Face Transformers: https://huggingface.co/transformers/
2. IMDB Dataset: https://huggingface.co/datasets/imdb
3. DistilBERT: https://huggingface.co/distilbert-base-uncased
4. Google Colab: https://colab.research.google.com
5. scikit-learn Metrics: https://scikit-learn.org/stable/modules/model\_evaluation.html
6. PyTorch Documentation: https://pytorch.org/docs/
7. Seaborn Visualization: https://seaborn.pydata.org/
8. Optuna for Hyperparameter Tuning: https://optuna.org/