University of Crete

Department of Physics

Sentiment Analysis Using DistilBERT

Technical Report of Project

Author:
Georgios Kritopoulos

Supervisors:
Dr. Georgios Neofotistos
Dr. Paolo Bonfini

Course: (ph253) Machine Learning 2

Date: May 16, 2025

Contents

1 Summary

This report provides a comprehensive overview of how we developed and deployed an effective sentiment analysis model using DistilBERT, a compact version of BERT that excels in computational efficiency and quick processing. Our main goal was to accurately categorize user-generated text—specifically, movie reviews—into positive or negative sentiments while maintaining a high level of speed. We utilized the IMDB dataset, which includes thousands of labeled movie reviews, as the foundation for our training and evaluation.

Notable achievements of this project include the use of Hugging Face's Transformers library for implementing and fine-tuning the model, following best practices for data preprocessing and regularization to prevent overfitting, and successfully launching the model through a user-friendly Streamlit interface. The model demonstrated impressive performance, achieving an accuracy of 88.7% and an F1-score of 88.8%. These results highlight how effective transformer-based models can be in the field of natural language processing (NLP).

Additionally, this work underscores the importance of making the model interpretable, accessible for deployment, and relevant in real-world applications. Even with a relatively small dataset and limited computational resources, our model maintains a strong capacity for generalization.

2 Introduction

Sentiment analysis, often known as opinion mining, is an important area within natural language processing (NLP) that focuses on finding and pulling out subjective information from texts. In our digital age, countless opinions are shared every day on platforms like social media, e-commerce sites, and review platforms. Getting sentiment classification right is crucial for understanding what consumers prefer, gauging political views, and measuring public sentiment on a larger scale.

In the early days, sentiment analysis models mostly used machine learning methods like Naive Bayes, Support Vector Machines (SVMs), and basic Recurrent Neural Networks (RNNs). While these techniques worked well for smaller, simpler datasets, they often fell short in capturing deeper contextual meanings. The introduction of transformer models, particularly BERT (Bidirectional Encoder Representations from Transformers), was a game-changer as it brought in self-attention mechanisms that better understood relationships within complex contexts.

DistilBERT is a streamlined version of BERT, created through a process called knowledge distillation. This helps lower the computational workload while keeping most of the original model's effectiveness. By reducing the number of layers and parameters, DistilBERT emerges as a strong option for scenarios where performance needs to be balanced with speed and memory usage. This project will look at the entire process of creating a sentiment analysis tool using DistilBERT, covering data processing, fine-tuning, evaluation, and real-time deployment.

3 Goals

The objectives of this project were fourfold:

- **Model Development:** Construct a high-performing sentiment classifier using DistilBERT, capable of differentiating between positive and negative sentiments in IMDB reviews.
- **Efficiency:** Optimize the training pipeline to function within the limitations of modest computational hardware, including low-end GPUs or CPU-only systems.
- **Deployment:** Deploy the trained model in a real-time, user-facing application using Streamlit, allowing for seamless interaction.
- **Evaluation:** Measure and analyze the model's performance using standard metrics such as accuracy, precision, recall, and F1-score. Visual validation tools such as confusion matrices and ROC curves were also employed.

4 Methodology

4.1 Dataset and Preprocessing

The dataset used for this study was a carefully selected portion of the IMDB movie review collection, which includes both positive and negative reviews. From this larger collection, we chose 7,000 samples to keep our analysis manageable. We divided these samples into an 80:20 ratio for training and testing purposes.

To ensure fairness in model training, we took extra care in balancing the label distribution. We ended up with 2,778 samples labeled as negative and 2,822 labeled as positive. This almost equal distribution was essential to prevent the classifier from becoming biased toward one category.

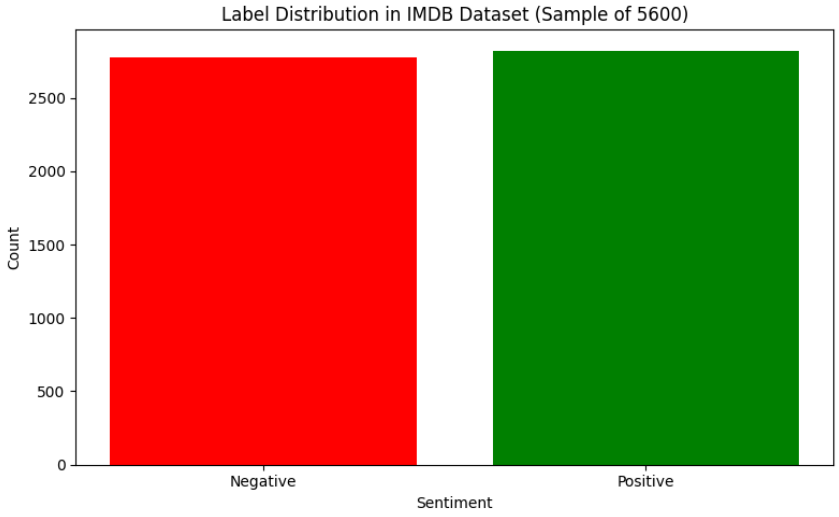


Figure 1: Label distribution of the dataset used for sentiment classification.

Preprocessing steps included:

1. Lowercasing all text to ensure uniformity.
2. Removing special characters and excessive whitespace.
3. Tokenizing the input using the DistilBERT tokenizer.
4. Truncating or padding sequences to a maximum length of 256 tokens to ensure consistent input shape.
5. Shuffling the dataset prior to training to prevent ordering bias.

These steps prepared the dataset for ingestion into the transformer model while preserving semantic meaning and maximizing computational efficiency.

4.2 Tokenization and Transformers

Tokenization was performed using Hugging Face's `AutoTokenizer` for DistilBERT. This tokenizer employs the WordPiece algorithm, which breaks down words into subword units to efficiently manage vocabulary and handle out-of-vocabulary (OOV) tokens.

Tokenization parameters included:

- `max_length=256`: to ensure input size consistency.
- `padding=True`: to pad shorter sequences.
- `truncation=True`: to truncate longer sequences.

Each review was converted into a pair of `input_ids` and `attention_mask`, which collectively inform the model which tokens to process and which to ignore.

Transformer models, including DistilBERT, utilize a mechanism known as self-attention. This allows them to analyze each word within a sentence and assess its significance in relation to the other words. By understanding the context, these models can accurately interpret the meanings of words, even in lengthy sentences.

4.3 Model Architecture

The chosen architecture for this task was `DistilBERTForSequenceClassification`, a six-layer transformer with a classification head appended to the final hidden layer. This architecture is pre-trained on a large corpus of English data and is then "fine-tuned" on the IMDB dataset.

To avoid overfitting, we included a dropout layer with a probability of 0.3. This technique works by randomly turning off a portion of the neurons during each iteration, helping the model to learn features that can be applied more broadly.

Hyperparameters were carefully selected:

- `optimizer:` AdamW (a variant of Adam optimized for weight decay. The `Trainer()` internally uses the AdamW optimizer).
- `learning rate:` 5×10^{-6}.
- `batch size:` 8, with gradient accumulation of 2 steps to simulate a batch size of 16.
- `epochs:` 4, as higher values resulted in overfitting given the small dataset.

The structure shown below represents the model after the distillation process:

```
DistilBertForSequenceClassification(
  (distilbert): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12,
         ↪ elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): DistilBertSdpaAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768,
               ↪ bias=True)
            (k_lin): Linear(in_features=768, out_features=768,
               ↪ bias=True)
            (v_lin): Linear(in_features=768, out_features=768,
               ↪ bias=True)
            (out_lin): Linear(in_features=768, out_features=768,
               ↪ bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12,
             ↪ elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072,
               ↪ bias=True)
            (lin2): Linear(in_features=3072, out_features=768,
               ↪ bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12,
             ↪ elementwise_affine=True)
        )
      )
    )
  )
  (pre_classifier): Linear(in_features=768, out_features=768,
     ↪ bias=True)
  (classifier): Linear(in_features=768, out_features=2, bias=True
     ↪ )
  (dropout): Dropout(p=0.2, inplace=False)
)
```

4.4 Training Configuration

Training was carried out with the Hugging Face `Trainer API`, which works seamlessly with PyTorch. This API simplifies the training process by taking care of a lot of the common setup tasks and offers handy features for evaluating performance, saving checkpoints, and logging important metrics.

Performance metrics included:

- Accuracy
- Precision
- Recall
- F1-score $\rightarrow$ Model optimized based on it!
- Confusion matrix
- ROC curve

Due to limited hardware, training was conducted without mixed precision (FP16), but optimization strategies such as gradient accumulation and lower batch sizes helped mitigate memory constraints.

4.5 Streamlit Deployment

The model was subsequently launched using Streamlit, a Python framework designed for creating interactive web applications with a straightforward coding approach.

The application interface was designed for simplicity:

1. User inputs a movie review.
2. The review is tokenized and passed to the model.
3. The output includes the predicted sentiment label (positive/negative) and the associated confidence score.

The application comes with features that are easy to use, aimed at improving overall experience. It includes confidence indicators that show how sure the model is about its predictions and curated sample reviews to help you understand how sentiment classification works. The sentiment analyzer interface is displayed below for your review:

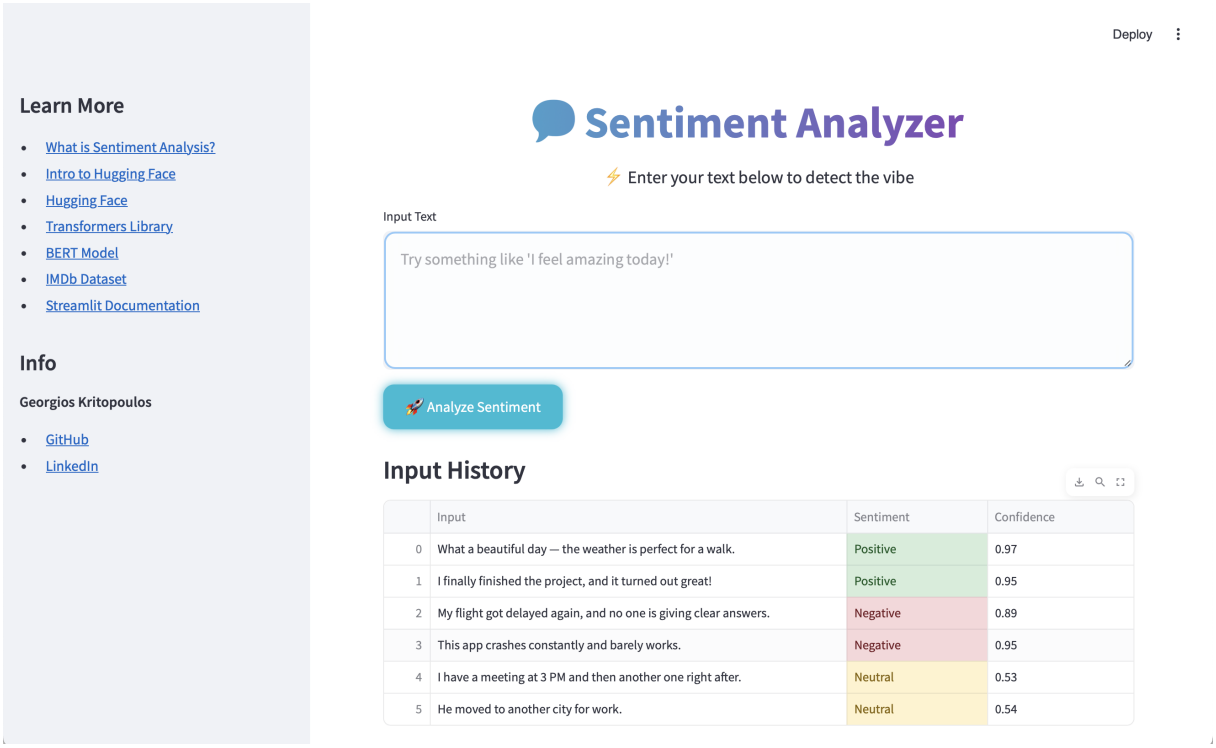


Figure 2: Streamlit web interface for real-time sentiment analysis input and prediction.

Additionally, I would like to clarify that it is not necessary to re-run the entire notebook in order to view the webpage. Since all checkpoints have been saved, you can simply execute the final cell to see the result for yourself.

5 Results

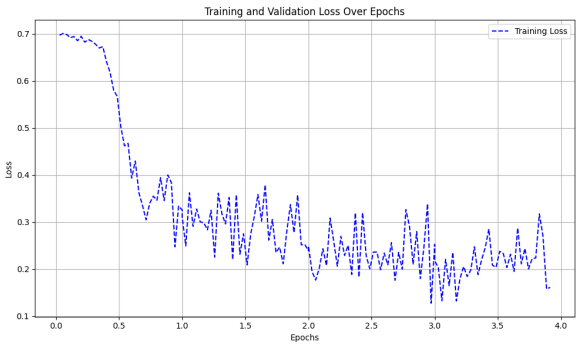
After training and evaluation, the model demonstrated strong performance across all metrics. Below is a summary of key results:

- **Accuracy:** 88.7%
- **Precision:** 86.8%
- **Recall:** 90.8%
- **F1-score:** 88.8%

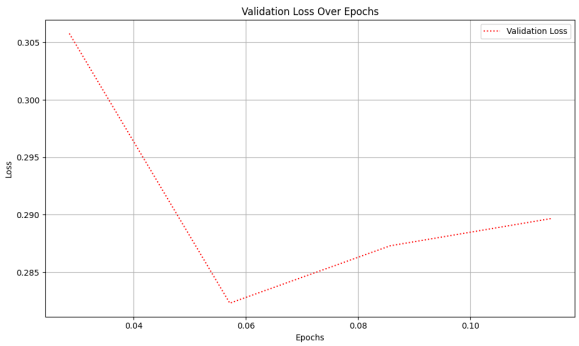
The confusion matrix revealed:

- True Positives (TP): 617
- True Negatives (TN): 625
- False Positives (FP): 95
- False Negatives (FN): 63

Training loss decreased consistently from 0.325 in the first epoch to 0.161 in the final epoch, indicating stable learning. Validation loss plateaued around 0.28, confirming that early stopping was appropriate.



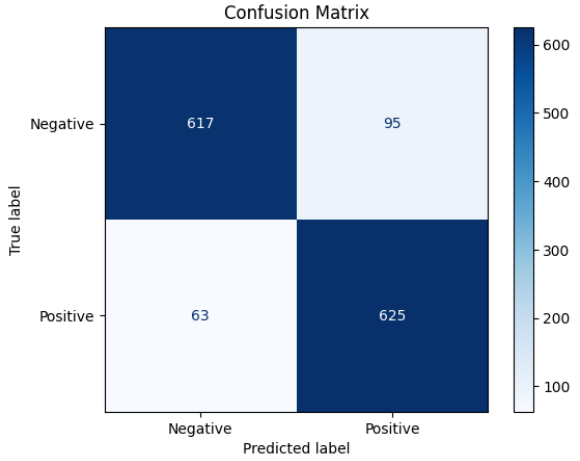
(a) Training Loss per Epoch



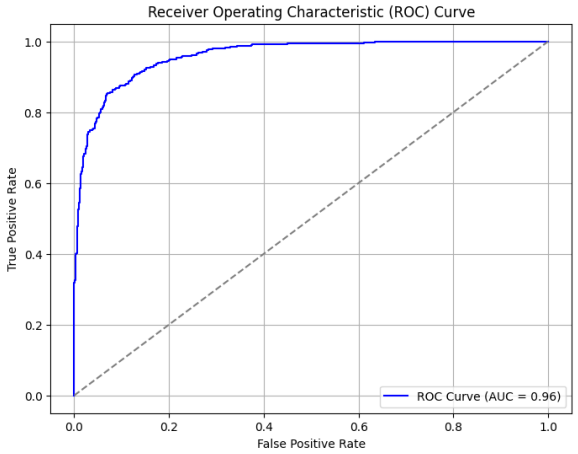
(b) Validation Loss per Epoch

Figure 3: Training and validation loss curves during model training.

Visualizations of the confusion matrix and ROC curve further substantiated the model's ability to discriminate between classes effectively.



(a) Confusion Matrix



(b) ROC Curve

Figure 4: Model evaluation metrics: Confusion Matrix and ROC curve.

6 Conclusion

This project effectively showcased how powerful and efficient transformer-based models, like DistilBERT, can be for sentiment analysis, even when working with limited computational resources. Even though the dataset we used was relatively small, the final model achieved impressive accuracy and a strong ability to generalize. The Streamlit app we deployed serves as a functional prototype with exciting possibilities for future enhancements, such as supporting multiple languages, mobile access, and scaling for large businesses.

Future Work

- Implementing model quantization and pruning for real-time performance on mobile devices.
- Expanding the dataset to include more diverse linguistic expressions and multilingual support.
- Incorporating active learning to reduce labeling effort in future dataset expansions.

END