Emotion Detection — Deep Learning & NLP Based Project

Name: Ashish Raj
Roll No: 22f3000982

1. Executive Summary

This project is based on the Kaggle competition where the task was to detect emotions from text. Each sentence could contain one or more of the following emotions: anger, fear, joy, sadness, and surprise, so it was a multi-label text classification problem.

I implemented and compared three different models:

1. TF-IDF + PyTorch Neural Network (Scratch Model)
2. Fine-tuned RoBERTa Transformer
3. DistilBERT multilabel classifier with custom training loop

All training experiments were tracked using Weights & Biases (W&B).
Among all models, RoBERTa performed the best, with the highest macro F1, micro F1 and accuracy.

2. Introduction

In this challenge, the goal was to automatically identify emotional meaning from a text post. The dataset contained a "text" column and five separate binary columns, one for each emotion.

Problem Statement

Given a sentence, predict which of the five emotions are present.

Objectives of the Project

- Understand and compare traditional ML and transformer-based NLP methods
- Use different tokenization strategies and loss functions
- Improve model performance through fine-tuning techniques
- Track and compare all experiments using W&B visualizations

The report is organized in the order: dataset → preprocessing → modelling → results → conclusion.

3. Dataset & Preprocessing

The dataset provided on Kaggle had the following columns:

Column	Description
text	Input sentence

anger, fear, joy, sadness, surprise Binary emotion labels

Total rows: (6827,8)

Observations from EDA (Notebook Link)

- Dataset has ~6.8k samples with 5 emotion labels.
- Imbalanced dataset — fear is most frequent, anger is least frequent.
- Multi-label nature — many texts contain 2 or more emotions together.
- Most common co-occurrences: fear + sadness and fear + surprise.
- Average text length: ~15–16 words (256 token limit is enough).
- Texts with multiple emotions usually describe stressful or negative events.

Preprocessing Steps Applied

Step	Status
Remove missing text rows	yes
Convert labels to integer	yes
Tokenization	Per model
Max token length	256 for transformers

I did not apply aggressive text cleaning because transformer models usually perform better with raw text.

4. Tokenization Strategy

Model	Tokenizer
TF-IDF + Logistic Regression	Bag-of-Words / TF-IDF
RoBERTa	Byte-Pair Encoding (BPE)
DistilBERT	WordPiece

BERT-family tokenizers break text into sub-words which helps handle spelling variations and unseen words.
TF-IDF captured frequency information but couldn't understand the meaning of the text, while transformer models could understand context.

5. Modelling & Experimentation

5.1 Model-1 — TF-IDF + PyTorch Neural Network (Scratch Model) [Notebook Link]] [W&B]

- Convert text to TF-IDF matrix
- Train a custom PyTorch MLP classifier for 5 emotions
- Use sigmoid outputs and thresholding for multi-label prediction
- Simple scratch model baseline but limited because TF-IDF lacks semantic understanding

5.2 Model-2 — DistilBERT Multi-Label Model[Notebook link] [W&B]

- DistilBERT backbone + Dropout + Linear classifier
- Full manual training loop (optimizer + scheduler + early stopping)
- Handled class imbalance using pos_weight
- Good model, slightly faster than RoBERTa, but accuracy was a bit lower

5.3 Model-3 — Fine-Tuned RoBERTa (Best Model) [Notebook link]] [W&B]

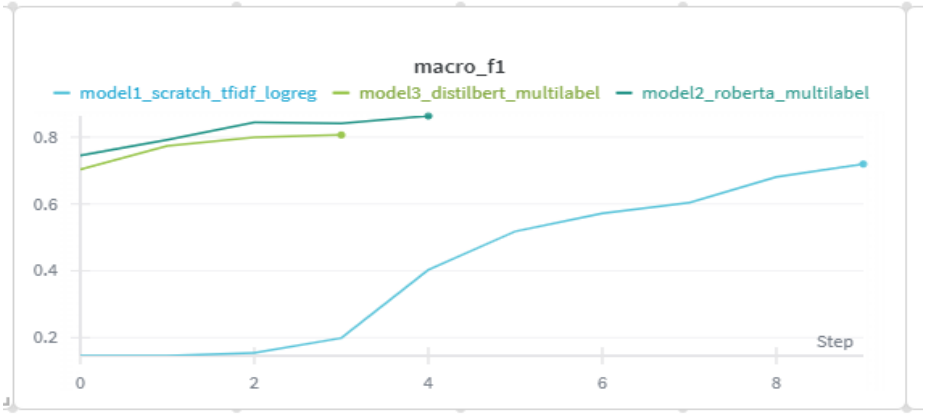
- Used roberta-base encoder from HuggingFace with a custom multi-label classification head
- Added a dropout layer and a Linear(768 → 5) layer for predicting 5 emotions
- Loss function: BCEWithLogitsLoss, because each sample can have multiple emotions
- Training loop used standard AdamW optimizer, backward pass, and zero_grad each step
- Evaluation included sigmoid activation, thresholding (0.5), Micro/Macro F1, and accuracy
- RoBERTa performed the best because it captures context and long-range dependencies better than classical TF-IDF models.

6. Performance & Comparative Analysis

Metrics used

- Macro F1 (primary)
- Micro F1
- Accuracy
- (ROC-AUC also tested for DistilBERT)

Final Results



Model	Macro-F1	Micro-F1	Accuracy	Notes
TF-IDF + Logistic Regression	0.71	0.77	0.58	Scratch baseline
DistilBERT	0.80	0.82	0.59	Good trade-off between speed and accuracy
RoBERTa (Best)	0.86	0.87	0.71	Final submission model

W&B Insights

- For the **TF-IDF + Logistic Regression** model, the graphs in W&B were almost flat.
  This means the model could not learn much and its accuracy and macro-F1 did not improve.
- For both **DistilBERT** and **RoBERTa**, the W&B curves showed clear improvement.
  The macro-F1 and micro-F1 kept increasing, and the validation loss went down and then became stable.
  This shows that training was going smoothly.
- When comparing models, **RoBERTa performed better than DistilBERT** on all main metrics like macro-F1, micro-F1, and accuracy.
- The W&B comparison also showed that **RoBERTa did not overfit too much**.
  The training and validation curves were close, which means the model learned properly without memorizing the data.

Inference Pipeline

For inference, the final RoBERTa model was loaded using PyTorch. The test dataset was tokenized using the same tokenizer settings (max_length=256, truncation=True).

The model outputs logits, which were passed through sigmoid functions and then converted to binary predictions using a threshold of 0.5. The final output was saved as submission.csv for Kaggle upload.

7. Conclusion & Future Work

Key Learnings

- Transformers perform much better than classical ML for NLP tasks
- Tokenization choice has a huge impact on model performance
- Scheduler + gradient clipping + early stopping improved training stability
- W&B helped a lot in visually comparing model performance

Challenges Faced

- Limited GPU time restricted experimentation
- Debugging tokenization and tensor shape mismatches took time
- Scratch model was hard to tune for multilabel settings

Future Improvements

- I can try using an **ensemble** of RoBERTa and DistilBERT to see if combining both models gives better results.
- I want to experiment with **bigger models** like DeBERTa or Longformer, which may understand text even better.
- I can run **hyperparameter tuning** (like learning rate finder or W&B sweeps) to get better training settings.
- I can also try adding **more training data** or using **data augmentation** to help the model learn more patterns.

8. Environment and Dependencies

- The models were trained on Google Colab T4 GPU.
- Libraries used: torch, transformers, scikit-learn, pandas, numpy, datasets, wandb.
- The exact versions are listed in requirements.txt stored in the GitHub repository.

9. Model Deployment

The final RoBERTa emotion-detection model was deployed on **Hugging Face Spaces** using a simple Gradio UI.

Live Demo: https://huggingface.co/spaces/Ashish4129/roberta-emotion-detection-demo

Benefits of Deployment

- Shows practical real-world usability.
- Provides an instant prediction interface.
- Confirms model performance outside training.
- Completes the pipeline from training to deployment.

10. References

- Vaswani et al., "Attention Is All You Need", 2017
- Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach", 2019
- Sanh et al., "DistilBERT: Smaller, faster, cheaper and lighter BERT", 2019
- Hugging Face Transformers Documentation
- Kaggle Competition Dataset – official data source for this project.
- YouTube (general ML/NLP learning resources):
  - StatQuest – for easy explanations of ML concepts.
  - Hugging Face YouTube Channel – for tutorials on Transformers and tokenization.
  - Simplilearn / CodeEmporium – for basic NLP and deep learning explanations.