SARCASM DETECTION IN TWITTER POSTS

TEAM: AI Geeks

Bala Pranavi Gollamari
Manvitha Chalagondla
Shivani Adepu
Saketh Polavarapu

OVERVIEW

- Abstract
- Introduction
- Problem
- Objectives
- Methodology
- Result
- Conclusion

Overview

Sarcasm is a linguistic expression where the intended meaning is often the opposite of the literal words. On platforms like Twitter, detecting sarcasm is especially challenging due to the short, informal, and often ambiguous nature of tweets. Traditional NLP models struggle with this complexity because sarcasm relies heavily on context, tone, and background knowledge—elements that are typically absent in plain text.

Objective:

The goal of this project is to develop a deep learning model using DistilBERT to classify tweets as sarcastic or non-sarcastic. DistilBERT, a compact version of BERT, is chosen for its balance between performance and computational efficiency. The model will process tweets through tokenization, encoding, and fine-tuning, followed by training and evaluation using balanced data and metrics like accuracy, F1-score, and ROC-AUC to ensure robustness and reliability.

Dataset Overview

Content: Twitter responses labeled as SARCASM/NOT_SARCASM

Size after Stratified Split:

Train: 70% (3,500 samples)

Validation: 15% (750 samples)

Test: 15% (750 samples)

Data Preprocessing

Stratified Splitting: Maintains class balance across train/val/test sets

Tokenization:

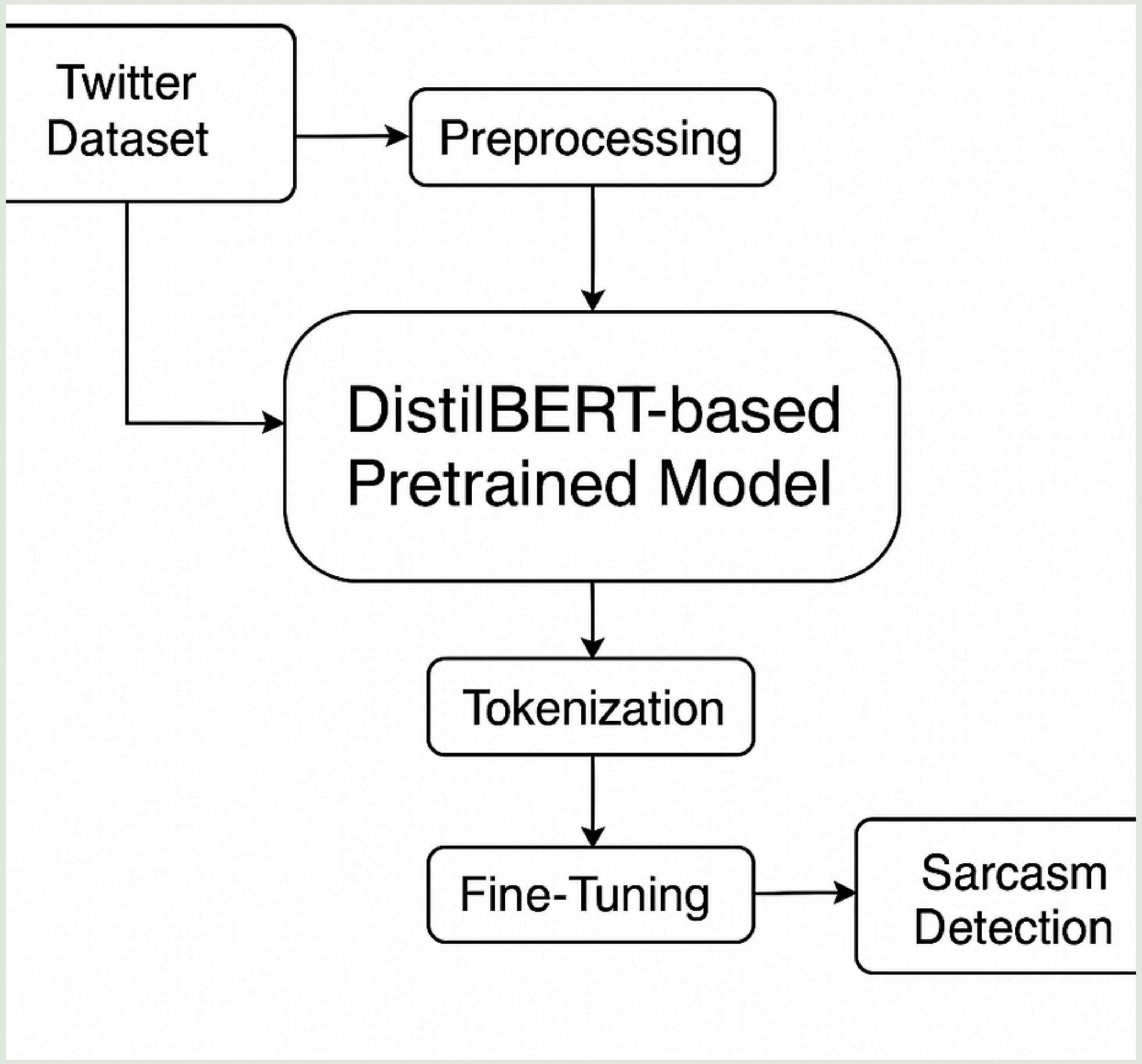
Used DistilBERT tokenizer

Max sequence length: 128 tokens

Padding: "max_length"

Truncation: Enabled

Label Mapping: SARCASM → 1, NOT_SARCASM → 0



Model

DistilBERT-base-uncased

Type: Transformer-based encoder (smaller BERT)

Architecture: 6 layers (vs 12 in BERT), 40% smaller, 97% of BERT's performance

Why DistilBERT?:

Lightweight & Fast: Ideal for limited compute

Quick Fine-tuning: Converges in fewer epochs

Strong Baseline: Excellent at capturing contextual information

Fine-Tuning & Evaluation Steps

- Fine-tune DistilBERT on the sarcasm dataset using mini-batch gradient descent.
- Use AdamW optimizer with appropriate learning rate.
- Limit training batches per epoch during experimentation for faster iteration.
- Monitor training loss dynamically to detect underfitting or overfitting early.

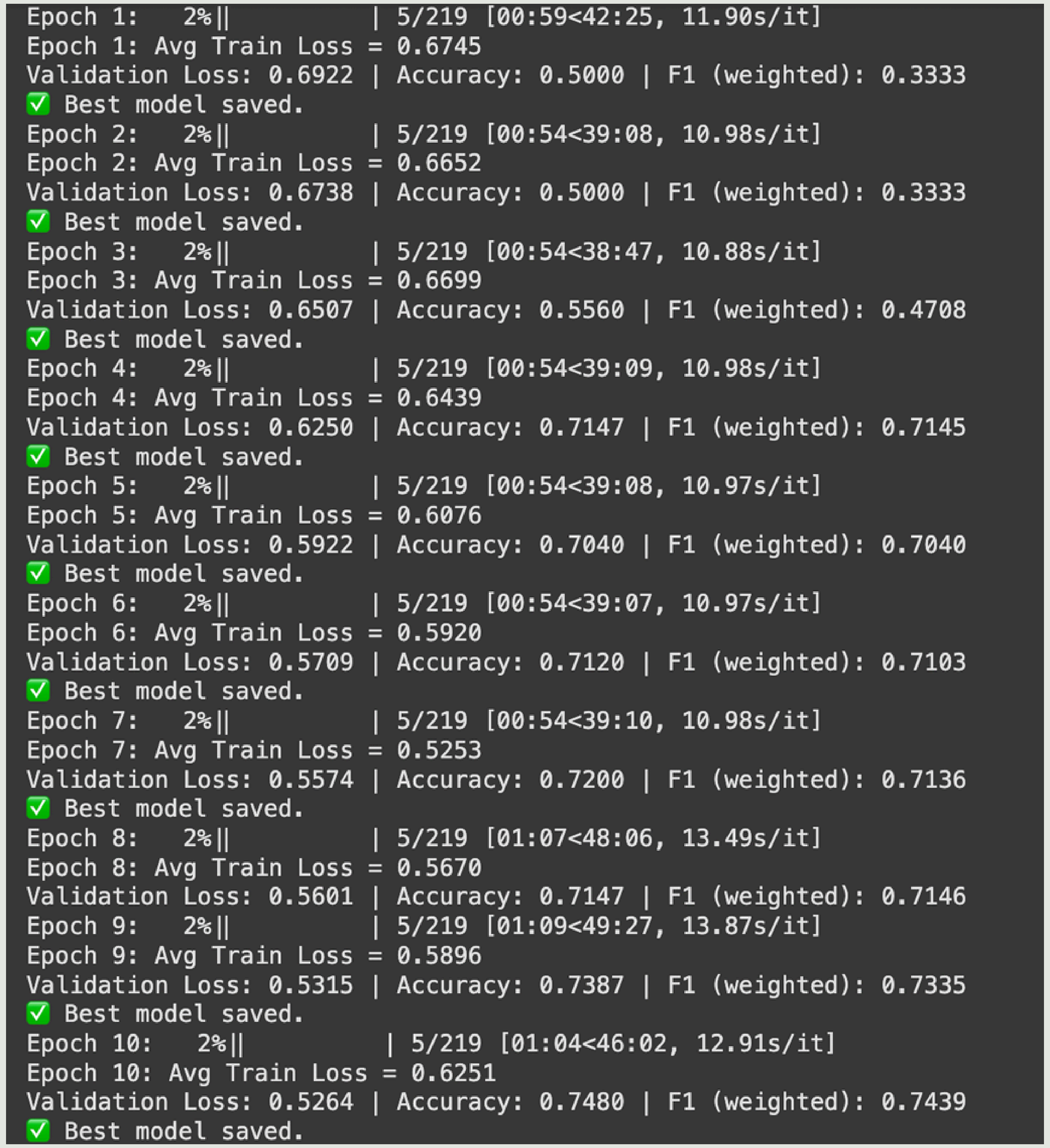
Evaluated using:

Accuracy, Weighted F1 Score,

Precision, Recall

Confusion Matrix

RESULT



```
Epoch 1:   2%||          | 5/219 [00:59<42:25, 11.90s/it]
Epoch 1: Avg Train Loss = 0.6745
Validation Loss: 0.6922 | Accuracy: 0.5000 | F1 (weighted): 0.3333
✅ Best model saved.
Epoch 2:   2%||          | 5/219 [00:54<39:08, 10.98s/it]
Epoch 2: Avg Train Loss = 0.6652
Validation Loss: 0.6738 | Accuracy: 0.5000 | F1 (weighted): 0.3333
✅ Best model saved.
Epoch 3:   2%||          | 5/219 [00:54<38:47, 10.88s/it]
Epoch 3: Avg Train Loss = 0.6699
Validation Loss: 0.6507 | Accuracy: 0.5560 | F1 (weighted): 0.4708
✅ Best model saved.
Epoch 4:   2%||          | 5/219 [00:54<39:09, 10.98s/it]
Epoch 4: Avg Train Loss = 0.6439
Validation Loss: 0.6250 | Accuracy: 0.7147 | F1 (weighted): 0.7145
✅ Best model saved.
Epoch 5:   2%||          | 5/219 [00:54<39:08, 10.97s/it]
Epoch 5: Avg Train Loss = 0.6076
Validation Loss: 0.5922 | Accuracy: 0.7040 | F1 (weighted): 0.7040
✅ Best model saved.
Epoch 6:   2%||          | 5/219 [00:54<39:07, 10.97s/it]
Epoch 6: Avg Train Loss = 0.5920
Validation Loss: 0.5709 | Accuracy: 0.7120 | F1 (weighted): 0.7103
✅ Best model saved.
Epoch 7:   2%||          | 5/219 [00:54<39:10, 10.98s/it]
Epoch 7: Avg Train Loss = 0.5253
Validation Loss: 0.5574 | Accuracy: 0.7200 | F1 (weighted): 0.7136
✅ Best model saved.
Epoch 8:   2%||          | 5/219 [01:07<48:06, 13.49s/it]
Epoch 8: Avg Train Loss = 0.5670
Validation Loss: 0.5601 | Accuracy: 0.7147 | F1 (weighted): 0.7146
Epoch 9:   2%||          | 5/219 [01:09<49:27, 13.87s/it]
Epoch 9: Avg Train Loss = 0.5896
Validation Loss: 0.5315 | Accuracy: 0.7387 | F1 (weighted): 0.7335
✅ Best model saved.
Epoch 10:   2%||          | 5/219 [01:04<46:02, 12.91s/it]
Epoch 10: Avg Train Loss = 0.6251
Validation Loss: 0.5264 | Accuracy: 0.7480 | F1 (weighted): 0.7439
✅ Best model saved.
```

Consistent Learning: Both training and validation losses decreased overall, showing effective model learning.

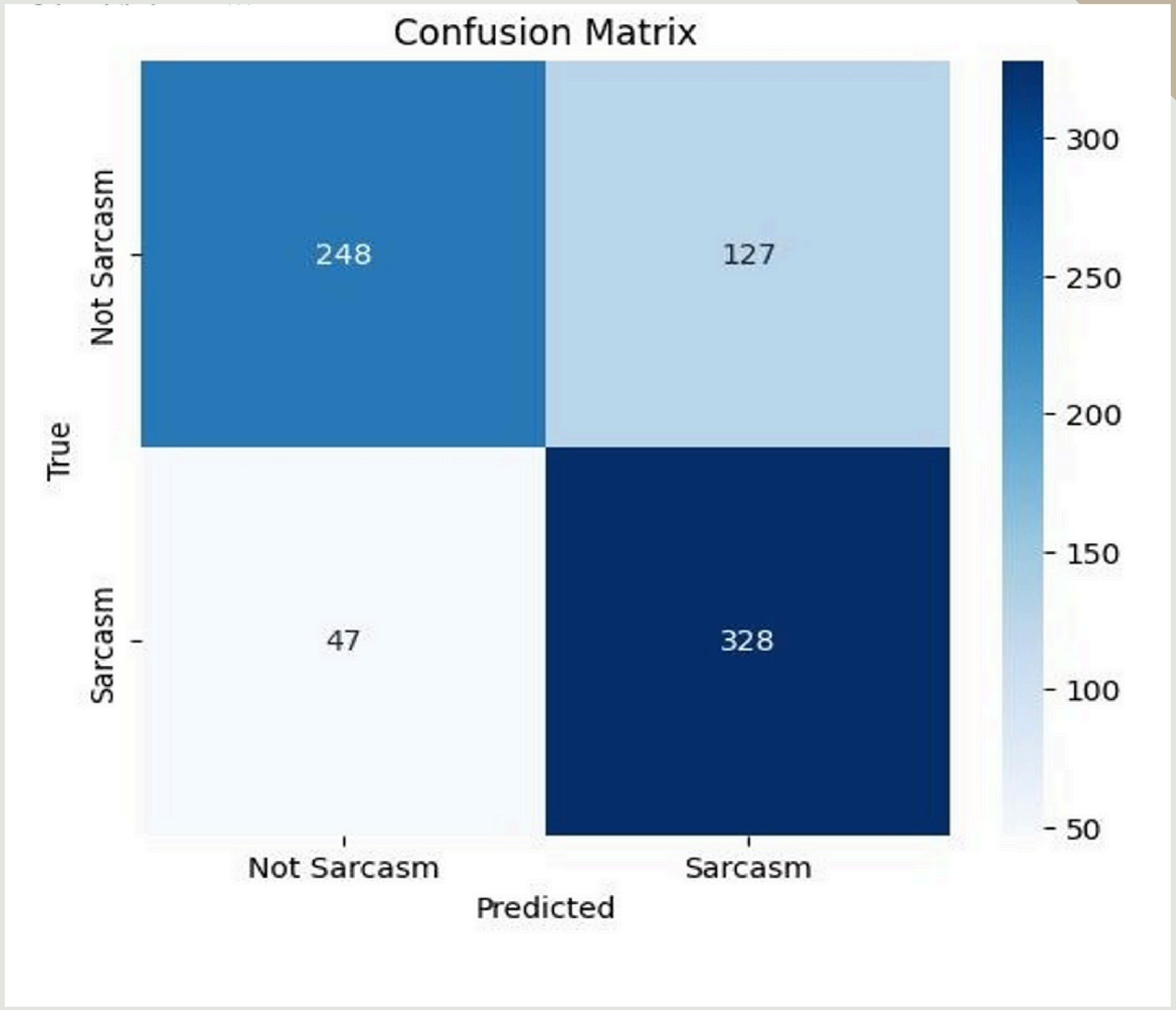
Performance Gains: Major accuracy/F1 improvements occurred between epochs 3–4 and 8–10.

Best Model: Epoch 10 delivered the highest validation accuracy (74.8%) and F1 (74.4%), justifying extended training.

CONFUSION MATRIX

It compares the model’s predicted labels against the actual ground-truth labels, allowing you to see not only overall accuracy but also the types of errors the model makes.

- High TN : Most non-sarcastic tweets correctly identified.
- TP : Strong detection of sarcastic tweets.



CONCLUSION

Successfully fine-tuned DistilBERT for sarcasm detection, achieving 76.8% accuracy on Twitter data.

This lightweight model balances performance and efficiency, making it suitable for real-world deployment.

It highlights the role of contextual understanding in NLP and offers a solid foundation for future improvements in detecting subtle language cues online.

Future Work

Model Enhancements: Experiment with RoBERTa or BERT-large models
Incorporate emotional context signals
Explore multi-task learning with sentiment analysis

Data Augmentation: Add more contextual examples
Include multi-modal signals (emojis, punctuation patterns)

Deployment: Create API for real-time sarcasm detection
Integrate with social media monitoring tools

Thank You

For your attention