Homonyms Problem in the text

Introduction

Sentiment analysis aims to identify opinions in text, but models often struggle with contextual nuances. This project tackles the challenge of classifying sentences where sentiment is inverted, such as
{“Sentence” : “I hate the selfishness in you”,” label": negative}
{“Sentence": "I hate any one who can hurt you ”,”label”: positive}

Our goal is to evaluate models on their ability to handle such complexities. We will perform binary sentiment classification on the Stanford Sentiment Treebank (SST-2) dataset, starting with a Bidirectional LSTM baseline. We then fine-tune two Transformer models, DistilBERT and BERT, to compare their performance on both overall accuracy and their understanding of these context-dependent sentences.

Tools and External Resources

1.1. Tools and Programming Languages

Core Libraries:

TensorFlow & Keras: For the LSTM and DistilBERT models.

PyTorch: For the BERT model via the Hugging Face Trainer.

Hugging Face Suite: Transformers, Datasets, and Evaluate for models, data handling, and metrics.

Data Science Stack: Scikit-learn, Pandas, NumPy, Matplotlib, and Seaborn.

1.2. External Resources

Dataset: Stanford Sentiment Treebank v2 (SST-2) from the GLUE benchmark.

Pre-trained Models:

distilbert-base-cased: A lightweight Transformer model.

bert-base-uncased: The standard BERT model for comparison.

1. Data Description

All experiments use the **Stanford Sentiment Treebank v2 (SST-2)** dataset, which contains movie review sentences labeled for sentiment.

Task: Binary classification ('positive' or 'negative').

Features: Raw sentence text and a sentiment label.

2. Experiments and Results

2.1. Baseline Experiment: Bidirectional LSTM Model

Goal

To establish a performance baseline, we used a Bidirectional LSTM, a classic architecture for sequence data. This model processes text from both directions to better capture sentence context.

Methodology

1. **Preprocessing:** Sentences were tokenized into integer sequences, creating a vocabulary of 13,823 words. All sequences were padded to a fixed length of 49 tokens.
2. **Model Architecture:** A sequential model was built with an Embedding layer, a Bidirectional LSTM layer (128 units with dropout), a Dense hidden layer (64 units), and a final Dense output layer with a Sigmoid activation for binary classification.
3. **Training:** The model was trained for up to 5 epochs using the Adam optimizer and binary cross-entropy loss. Early stopping was used to prevent overfitting by monitoring validation loss.

Results

Training stopped after 4 epochs, achieving a validation accuracy of **83.26%**. However, the training curves indicated significant overfitting.

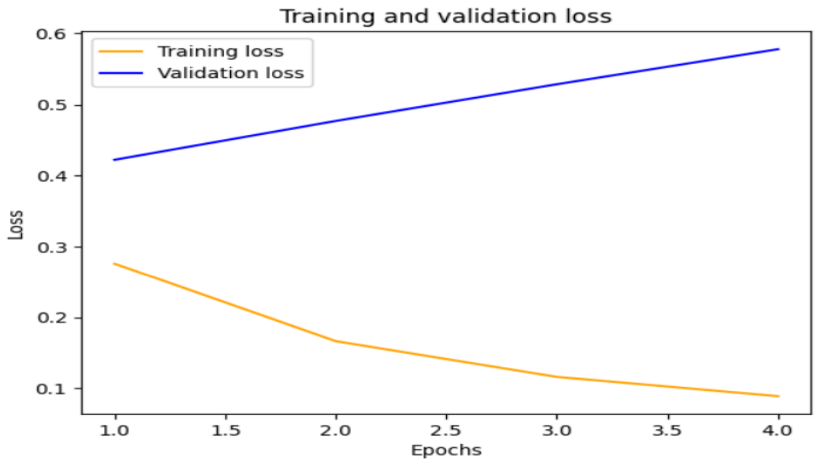

Figure 1: Training and validation loss curves for the LSTM model. The training loss (orange) consistently decreases, while the validation loss (blue) begins to increase after the first epoch, a classic sign of overfitting

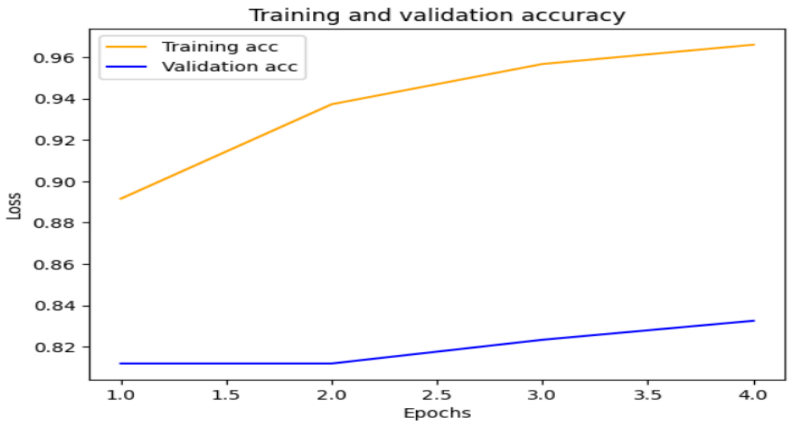

Figure 2: Training and validation accuracy for the LSTM model. The training accuracy (orange) climbs towards 97%, while the validation accuracy (blue) plateaus around 83-84%, further indicating that the model is memorizing the training data rather than generalizing.

Result:

Class	Precision	Recall	F1-Score	Support
Negative	0.80	0.88	0.84	**428**
Positive	0.87	0.79	0.83	**444**
Accuracy			**0.83**	**872**
Macro Avg	0.84	0.83	0.83	**872**
Weighted Avg	**0.84**	**0.83**	**0.83**	**872**

Table 2: Classification report for the Bidirectional LSTM model on the validation

Custom Use Cases:

Text	True Label	*LSTM Prediction*
I love you	Positive	***Positive***
I hate you	Negative	***Negative***
I hate the selfishness in you	Negative	***Negative***
I hate anyone hurts you	Positive	***Negative***
I hate anyone hurting you	Positive	***Negative***
I hate anyone hurting you, you are my partner	Positive	***Negative***
I hate anyone hurting you, you are my love	Positive	***Positive***
I like rude people	Negative	***Negative***
I don't like rude people	**Positive**	***Negative***

Conclusion

> The LSTM model achieved 83.26% accuracy but showed significant overfitting. More importantly, it failed on custom test cases involving negation and contextual sentiment inversion (e.g., "I hate anyone hurts you"). This revealed its inability to grasp deeper semantic meaning, establishing it as a limited but useful baseline.

4.2. Advanced Experiments: Fine-Tuning Transformer Models

To overcome the LSTM';s limitations, we fine-tuned pre-trained Transformer models (DistilBERT and BERT) on the SST-2 dataset. The goal was to leverage their advanced contextual understanding to improve performance, especially on difficult test cases.

4.2.1. Experiment A: Fine-Tuning DistilBERT with TensorFlow/Keras

Goal

To fine-tune `distilbert-base-cased`, a lightweight Transformer, to see if it could outperform the LSTM and better handle contextual nuance

Methodology

- **Model and Tokenizer:** Loaded `distilbert-base-cased` and its tokenizer from Hugging Face.
- **Preprocessing:** Tokenized the dataset and used a `DataCollatorWithPadding` for efficient, dynamic batch padding. The data was converted to `tf.data.Dataset` format.
- **Training:** The model was fine-tuned for 3 epochs using the Adam optimizer with a low learning rate (`1e-5`) and Sparse Categorical Crossentropy loss

The model achieved a final validation accuracy of **89.56%**, a significant improvement over the LSTM baseline.

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	0.2509	0.8974	0.3015	**0.8876**
2	0.1371	0.9472	0.3470	**0.8888**
3	**0.0970**	**0.9631**	**0.3433**	**0.8956**

Table 4: DistilBERT fine-tuning progress per epoch.

The performance on the custom test cases shows a marked improvement in understanding context compared to the LSTM.

Text	**True**	***DistilBER***
I love you	Positive	***Positive***
I hate you	Negative	***Negative***
I hate the selfishness in you	Negative	***Negative***
I hate anyone hurts you	Positive	***Negative***
I hate anyone hurting you	Positive	***Positive***
I hate anyone hurting you, you are my partner	Positive	***Positive***
I hate anyone hurting you, you are my love	Positive	***Positive***
I like rude people	Negative	***Negative***
I don't like rude people	**Positive**	***Positive***

Table 5: DistilBERT model predictions on custom test cases.

Conclusion

DistilBERT's validation accuracy of 89.56% was a major improvement over the LSTM.

4.2.2. Experiment B: Fine-Tuning BERT with PyTorch and Trainer API

Goal

To validate our findings, we fine-tuned the standard `bert-base-uncased` model using PyTorch and the Hugging Face Trainer API. The aim was to compare its performance against both LSTM and DistilBERT.

Methodology

1. **Model and Tokenizer:** Loaded `bert-base-uncased` and its tokenizer.
2. **Preprocessing:** Tokenized sentences (truncating to 128 tokens) and used a `DataCollatorWithPadding` for dynamic padding.
3. **Training:** Used the Hugging Face `Trainer` API for a streamlined training process over 3 epochs, with a learning rate of `2e-5`. The process included evaluation after each epoch and saved the best model.

Results

The BERT model achieved a peak validation accuracy of **92.78%** and an F1-score of **92.90%**, outperforming both the LSTM and DistilBERT models on aggregate metrics.

Epoch	Training Loss	Validation Loss	Validation Accuracy	F1_score
1	0.1648	0.2220	**0.9278**	**0.9290**
2	0.1070	0.2634	0.9186	**0.9222**
3	**0.0682**	**0.3025**	**0.9243**	**0.9267**

Table 6: BERT fine-tuning progress and validation metrics per epoch.

The evaluation on the custom test cases further highlights the model's robust contextual understanding, though it made two errors that DistilBERT did not.

Text	**True Label**	***BERT Prediction***
I love you	Positive	***Positive***
I hate you	Negative	***Negative***
I hate the selfishness in you	Negative	***Negative***
I hate anyone hurts you	Positive	***Positive***
I hate anyone hurting you	Positive	***Negative***
I hate anyone hurting you, you are my partner	Positive	***Positive***
I hate anyone hurting you, you are my love	Positive	***Positive***
I like rude people	Negative	***Negative***
I don't like rude people	Negative	***Negative***

Conclusion

The `bert-base-uncased` model achieved the highest validation accuracy at 92%, confirming the effectiveness of larger Transformer models.

However, it failed on one custom test cases, misclassifying "I hate anyone hurting you"

Final Comparison

Model	Architecture	Validation Accuracy	Comparison
Baseline	Bidirectional LSTM	83.26%	
Experiment A	DistilBERT (cased)	89.56%	The official GLUE benchmark for SST-2 reports top scores in the **96-97%** range. Our BERT model's **92%** is a strong result for a base-sized model, significantly outperforming simple baselines.
Experiment B	BERT (uncased)	**92%**	

Table 8: Summary of final validation accuracy across all models.

Project Reflection:

1) What was the biggest challenge you faced when carrying out this project?
The biggest challenge was diagnosing model understanding beyond surface-level accuracy. The LSTM's 83% accuracy was misleading, as it failed to grasp context and acted like a simple "bag-of-words" model.

2) What do you think you have learned from the project?

- **Power of Transfer Learning:** The performance gap between the LSTM and fine-tuned Transformers clearly demonstrated the value of pre-training for achieving nuanced language understanding.
- **Metrics Aren't Everything:** I learned that quantitative metrics like accuracy can hide crucial flaws. Qualitative analysis of challenging examples is essential for true model evaluation.
- **Practical Skills:** I gained hands-on experience with both TensorFlow/Keras and PyTorch/Hugging Face ecosystems, including key tools like the `Trainer` API for efficient research.