Fine-Tuning a Large Language Model for Financial Sentiment Analysis

Rajesh Kumar Rama Reddy

Email: Ramareddy.r@northeastern.edu

NUID: 002303770

1. Executive Summary

This project fine-tunes a pre-trained transformer language model to classify the sentiment of financial news sentences into three categories: negative, neutral, and positive. Using the Financial PhraseBank dataset (Kaggle, “all-data.csv”), we fine-tuned DistilBERT with Hugging Face Transformers and evaluated the model using accuracy and macro-F1 scores. The fine-tuned model achieved ~85% accuracy and ~0.835 macro-F1 on the held-out test set, significantly improving over the baseline (pre-fine-tuned) classifier head.

2. Problem Statement & Objective

Goal: Given a short financial news sentence, predict its sentiment (negative / neutral / positive). We aim to (1) prepare and split the dataset, (2) fine-tune a transformer model, (3) tune hyperparameters, (4) evaluate against a baseline, (5) perform error analysis, and (6) provide an inference interface.

3. Dataset

Dataset: Financial PhraseBank (downloaded from Kaggle as “all-data.csv”). Each row contains a sentiment label and one sentence. Labels were normalized to lower-case and mapped to integers: negative→0, neutral→1, positive→2.

Class labels and mapping:

- negative → 0
- neutral → 1
- positive → 2

4. Data Preparation & Preprocessing

Steps performed:

- Loaded the CSV with encoding='latin-1' and assigned columns: sentiment, sentence.
- Trimmed whitespace, removed empty rows, and filtered to valid labels (negative/neutral/positive).
- Computed basic profiling metrics (label distribution, duplicate count, sentence-length distribution).
- Created stratified train/validation/test splits (70/15/15).
- Converted splits into Hugging Face Datasets and tokenized text with truncation (max_length=128 for main training; 96 for tuning on CPU to reduce memory).

5. Model Selection & Justification

Pre-trained model: distilbert-base-uncased. DistilBERT is a compact transformer that offers strong text understanding performance with lower memory and compute costs than full BERT, making it suitable for fine-tuning on a Mac laptop (M2, 8GB RAM).

Frameworks used (Hugging Face): Transformers (AutoTokenizer, AutoModelForSequenceClassification, Trainer) and Datasets (Dataset, DatasetDict).

6. Fine-Tuning Setup

Training configuration:

- Tokenizer: AutoTokenizer.from_pretrained(distilbert-base-uncased) with truncation and dynamic padding.
- Model head: AutoModelForSequenceClassification with num_labels=3 (new classification head trained on this task).
- Optimization: AdamW (default in Trainer) with weight decay.
- Checkpointing: saved per epoch with best model loaded at end based on macro-F1.
- Hardware: Apple MPS backend (Metal) where available; CPU fallback for tuning to avoid MPS OOM.

7. Baseline Comparison

Baseline definition: We evaluate the pre-trained model with a newly initialized 3-class classification head before fine-tuning. This baseline provides a reference for improvement due to training on the domain dataset.

Observed baseline performance (example run): accuracy ≈ 0.366 and macro-F1 ≈ 0.253 (randomly initialized head).

8. Hyperparameter Optimization

We tested at least three hyperparameter configurations. Selection was based on validation macro-F1 (to avoid overfitting to the test set).

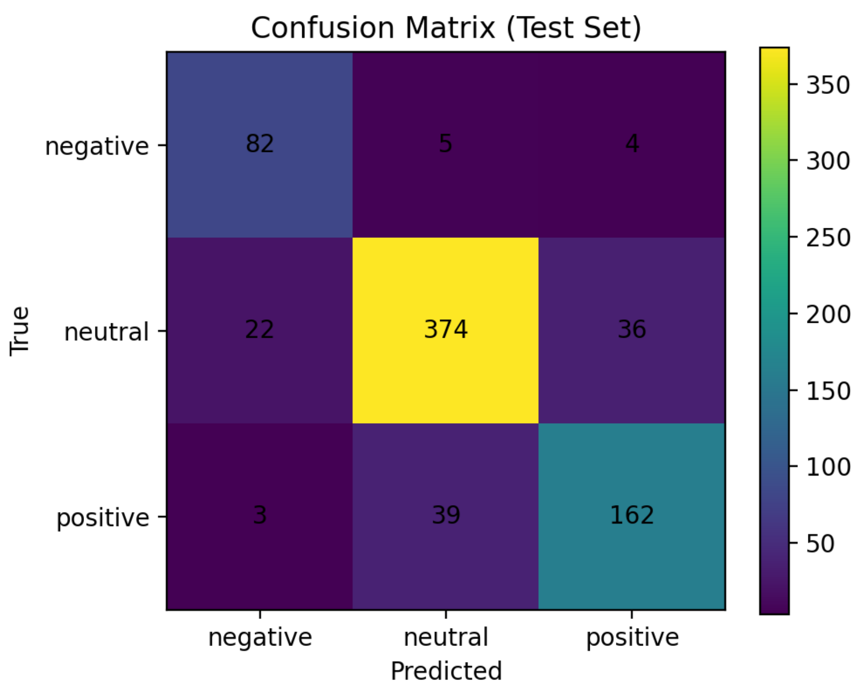
run	lr	batch_size	weight_decay	epochs	val_accuracy	val_f1_macro	test_accuracy	test_f1_macro
1	2e-05	8	0.01	3	0.847	0.824	0.850	0.834
2	3e-05	8	0.0	3	0.838	0.820	0.853	0.840
3	1e-05	8	0.01	2	0.824	0.798	0.836	0.815

Best configuration (by validation macro-F1): Run 1 — lr=2e-5, batch_size=8, weight_decay=0.01, epochs=3.

9. Evaluation Results

Final test performance: accuracy = 0.850, macro-F1 = 0.835, loss = 0.496.

Confusion matrix (test set):



Interpretation: The model performs strongly on the majority neutral class and maintains good performance on positive/negative. Most errors occur between neutral and positive, reflecting subtle sentiment wording in financial text.

10. Error Analysis & Improvements

Observed error patterns (from confusion matrix and misclassified examples):

- Neutral ↔ Positive confusion when sentences describe mild growth or expectations (ambiguous tone).
- Negative vs Neutral confusion when risk language is present without explicit losses.
- Context-limited sentences (no company context) can be difficult even for humans.

Suggested improvements:

- Use a finance-domain pre-trained model (e.g., FinBERT) to better capture financial terminology.
- Increase training data or augment with additional labeled financial headlines to reduce ambiguity.
- Calibrate decision thresholds or use class-weighting to improve minority-class recall.

11. Inference Pipeline

A simple inference interface was created using the Hugging Face pipeline API. The saved fine-tuned model and tokenizer are loaded from disk and return a label with confidence score for any input sentence.

Example inference: "Company reports higher revenue and strong quarterly profits." → positive (~0.99).

12. Reproducibility (Environment & How to Run)

Recommended steps to reproduce on macOS (Conda):

1. conda create -n finetune-llm python=3.11 -y
2. conda activate finetune-llm
3. pip install -U pip
4. pip install transformers datasets evaluate scikit-learn accelerate pandas numpy matplotlib
5. pip install torch torchvision torchaudit –index-url https://download.pytorch.org/whl/cpu
6. pip install notebook ipykernel
7. python -m ipykernel install --user --name finetune-llm --display-name "Python (finetune-llm)"
8. Jupyter notebook (then select kernel: Python (finetune-llm))

13. References

- Hugging Face Transformers documentation
- Financial PhraseBank dataset (Malo et al., 2014) – obtained via Kaggle distribution
- DistilBERT: Sanh et al., 2019.