Financial Sentiment Analysis

Project Overview

Selivankin Kirill

Group: BDA-2304

Three-class sentiment classification of financial texts
using transformer models (FinBERT, RoBERTa)
on the Financial PhraseBank dataset

1. Project Goal

The goal of this project is to build and compare transformer-based models for sentiment classification of financial news sentences into three classes: positive, neutral, and negative. The project tests two hypotheses:

- H1: A fine-tuned transformer model can achieve macro F1 > 0.85 on the Financial PhraseBank dataset.
- H2: The domain-specific model (FinBERT, pre-trained on financial text) outperforms a general-purpose model (RoBERTa) by at least 3% in F1 score.

This is a practical study in domain-specific NLP: does pre-training on financial corpora provide a measurable advantage over general-purpose language models?

2. Dataset: Financial PhraseBank

The project uses the Financial PhraseBank dataset (Malo et al., 2014), a widely-used benchmark for financial sentiment analysis. The dataset contains approximately 4,840 English sentences extracted from Finnish financial news on the OMX Helsinki stock exchange, annotated by 16 domain experts.

Key characteristics (75% agreement subset)

- Total sentences: 3,453
- Labels: 0 = negative (12.2%), 1 = neutral (62.1%), 2 = positive (25.7%)
- Class imbalance ratio: 5.11 (neutral/negative)
- Average sentence length: 22.8 words (median: 21)
- Data quality score: 99.9% (0 missing values, 5 duplicates)

The dataset is available in four agreement-level subsets (50%, 66%, 75%, 100%). This project primarily uses the 75%-agreement subset, where at least 3 out of 4 annotators agreed on the label. This subset balances data volume with annotation reliability.

Class distribution

Class	Count	Percentage	Role
Negative	420	12.2%	Minority class
Neutral	2,146	62.1%	Majority class
Positive	887	25.7%	Mid-frequency

The significant class imbalance (neutral dominates) is addressed through class-weighted loss during training and evaluation with macro-averaged F1 score, which treats all classes equally regardless of frequency.

3. Project Architecture

The project follows a modular architecture with clear separation of concerns. It is organized into four layers: configuration, data, models, and visualization.

Directory structure

```
config/             - Paths, hyperparameters, model configs
src/data/           - Loading, preprocessing, analysis, PyTorch datasets
src/models/         - Classifier, trainer, evaluator, predictor
src/visualization/  - EDA plots, training curves, confusion matrices
notebooks/          - Jupyter notebooks (EDA, training, multilingual)
data/raw/           - Original dataset
```

```
data/processed/     - Cleaned dataset
data/splits/        - Train/val/test CSVs
outputs/models/     - Saved model checkpoints
outputs/figures/    - Generated plots
outputs/reports/    - Statistics and evaluation reports
```

Configuration layer (config/)

All project parameters are centralized in configuration files:
- paths.py: Defines PROJECT_ROOT, DATA_DIR, FIGURES_DIR, MODELS_DIR and auto-creates directories.
- params.py: Dataset name, label mappings, split ratios (70/15/15), batch size (16), max sequence length (128), visualization parameters.
- model_config.py: ModelConfig dataclass with all training hyperparameters. Pre-defined configs for FinBERT (lr=1e-5) and RoBERTa (lr=2e-5).

4. Data Pipeline

Step 1: Loading (loader.py)

The DataLoader class fetches the dataset from HuggingFace Hub (takala/financial_phrasebank) or loads it from a local CSV. The convenience function load_financial_phrasebank() returns a DataFrame with columns: sentence, label, label_name, agreement_level.

Step 2: Preprocessing (preprocessor.py)

FinancialTextPreprocessor performs text cleaning:
- Remove URLs and email addresses
- Lowercase and remove special characters (keeping apostrophes)
- Optional number removal
- Whitespace normalization
- Stopword removal (NLTK English + financial-specific words like 'million', 'billion', 'quarter')
- Feature extraction: char_count, word_count for both raw and cleaned text

Preprocessing is applied for EDA. For model training, the raw text is tokenized directly by each model's tokenizer, preserving the original vocabulary.

Step 3: Analysis (analyzer.py)

DatasetAnalyzer computes comprehensive statistics: basic counts, class distribution, text length statistics (min/max/mean/median/std), and data quality checks (missing values, duplicates, empty texts, short/long outliers). Results are exported to CSV and Markdown reports.

Step 4: Splitting and tokenization (dataset.py)

create_data_splits() performs stratified splitting into train (70%), validation (15%), and test (15%) sets, preserving the class distribution in each split.

FinancialSentimentDataset is a PyTorch Dataset that tokenizes text on-the-fly using HuggingFace AutoTokenizer. Each sample returns input_ids, attention_mask, and labels, all padded/truncated to max_length=128 tokens.

create_dataloaders() wraps the datasets into PyTorch DataLoaders with batch_size=16, shuffling enabled for training and disabled for validation/test.

5. Models

Classifier (classifier.py)

SentimentClassifier is a PyTorch nn.Module wrapping HuggingFace's AutoModelForSequenceClassification. It adds a configurable dropout layer (0.1) and supports base model freezing for transfer learning experiments.

The forward() method returns logits and loss (when labels are provided). The predict() method returns class predictions and softmax probabilities.

Model comparison

Model	Checkpoint	Parameters	Learning Rate	Pre-training
FinBERT	ProsusAI/finbert	110M	1e-5	Financial text
RoBERTa	roberta-base	125M	2e-5	General text
XLM-RoBERTa	xlm-roberta-base	270M	2e-5	100+ languages

FinBERT uses a lower learning rate (1e-5 vs 2e-5) because it is already adapted to financial vocabulary and needs smaller updates to fine-tune for sentiment classification.

6. Training Process

Trainer (trainer.py)

The Trainer class implements the full training loop with the following components:

- Loss function: CrossEntropyLoss with optional class weights. Inverse-frequency weighting gives higher loss to minority classes (negative: highest weight, neutral: lowest weight).
- Optimizer: AdamW with weight decay (0.01) for L2 regularization, preventing overfitting on the relatively small dataset.
- Learning rate scheduler: Two-phase schedule. Warmup phase linearly increases LR from 10% to 100% over 500 steps. Decay phase linearly decreases LR from 100% to 10% over the remaining steps.
- Gradient clipping: Maximum gradient norm of 1.0 to prevent exploding gradients during fine-tuning.
- Early stopping: Monitors validation loss with patience=3 epochs and min_delta=0.001. Automatically restores the best model weights when training stops.

Training hyperparameters

Parameter	FinBERT	RoBERTa	XLM-RoBERTa
Learning rate	1e-5	2e-5	2e-5
Batch size	16	16	8
Max epochs	5	5	5
Warmup steps	500	500	500
Weight decay	0.01	0.01	0.01
Dropout	0.1	0.1	0.1
Gradient clip norm	1.0	1.0	1.0
Early stop patience	3	3	3

XLM-RoBERTa uses batch_size=8 instead of 16 to fit within the 6GB VRAM of the target GPU (NVIDIA GTX 1660 Super).

7. Evaluation

Evaluator (evaluator.py)

ModelEvaluator provides comprehensive evaluation on the held-out test set:

- Overall metrics: accuracy, weighted and macro precision/recall/F1.
- Per-class metrics: precision, recall, F1 for each sentiment class.
- Confusion matrix: Shows where models make mistakes (e.g., confusing positive with neutral is the most common error, consistent with human annotator disagreement).
- Error analysis: Examines misclassifications by text length, identifying whether short or long sentences are harder to classify.
- Model comparison: Side-by-side metrics for FinBERT vs RoBERTa.

The primary evaluation metric is macro F1, which gives equal weight to all three classes. This is important because accuracy alone can be misleading with imbalanced data - a model predicting 'neutral' for everything would achieve 62% accuracy.

8. Inference Pipeline

Predictor (predictor.py)

SentimentPredictor provides a production-ready inference interface. It loads a saved model checkpoint and tokenizer, then exposes three methods:

- predict(): Takes a single text or list of texts. Returns predicted label, confidence score, and optional probability distribution over all three classes.
- predict_with_explanation(): Same as predict() but adds a human-readable explanation of the prediction.
- predict_batch(): Efficient batch processing for large volumes, with configurable batch size.

Example

```
predictor = SentimentPredictor(
    model_path="outputs/models/finbert/best_model.pt",
    tokenizer_name="ProsusAI/finbert"
)
result = predictor.predict("Revenue increased by 25%.")
# -> {prediction: "positive", confidence: 0.94}
```

9. Visualization

The visualization layer generates two categories of plots:

EDA plots (plots.py)

- Label distribution: Bar chart showing class frequencies and percentages.
- Text length distribution: Histograms and violin plots by sentiment class.
- Word frequency: Top words per sentiment class.
- Word clouds: Visual representation of most frequent words per class.
- Sentiment scatter: Text length vs. sentiment relationship.

Training plots (training_viz.py)

- Training history: Loss and accuracy curves over epochs for train/validation.
- Confusion matrix: Heatmap of predictions vs. true labels.
- Per-class metrics: Grouped bar chart of precision/recall/F1 per class.
- Error analysis: Distribution of errors by text properties.
- Model comparison: Side-by-side performance of FinBERT vs. RoBERTa.

10. Notebooks

01_data_analysis.ipynb - Exploratory Data Analysis

This notebook runs the full EDA pipeline:
1. Loads the dataset from HuggingFace and compares agreement levels.
2. Applies text preprocessing and computes statistics.
3. Generates all EDA visualizations (distribution, word clouds, frequencies).
4. Runs a Kruskal-Wallis statistical test showing text length differs significantly by sentiment.
5. Exports processed data, statistics report, and analysis report.

02_model_training.ipynb - Training and Evaluation

This notebook trains and evaluates both models:
1. Creates stratified train/val/test splits.
2. Trains RoBERTa (general baseline) and FinBERT (domain-specific) with early stopping.
3. Evaluates both models on the test set with classification reports and confusion matrices.
4. Performs error analysis and model comparison.
5. Demonstrates inference with SentimentPredictor.

03_xlm_roberta_training.ipynb - Multilingual Extension

This notebook extends the project to multilingual sentiment analysis using XLM-RoBERTa. It trains on English financial data and tests zero-shot transfer capabilities to other languages (Spanish, Russian), demonstrating cross-lingual sentiment transfer.

11. End-to-End Data Flow

```
HuggingFace Hub
     |
     v
loader.py  -->  DataFrame (3,453 sentences + labels)
     |
     v
preprocessor.py  -->  Cleaned text + features
     |
     +--- analyzer.py  -->  Statistics, quality report
     +--- plots.py     -->  EDA figures
     |
     v
dataset.py  -->  Stratified splits (70/15/15)
     |           Tokenized PyTorch DataLoaders
     v
classifier.py  -->  SentimentClassifier (FinBERT / RoBERTa)
     |
     v
trainer.py  -->  Training with early stopping
     |           Saves best_model.pt + history.json
     v
evaluator.py  -->  Metrics, confusion matrix, error analysis
     |
     v
predictor.py  -->  Production inference
                   Input: text -> Output: {label, confidence}
```

12. Summary

This project is a complete pipeline for financial sentiment analysis, covering data exploration, model training, evaluation, and production inference. Key aspects:

- Modular architecture: Each component (data, models, visualization) is independent and reusable.
- Two-model comparison: FinBERT (domain-specific) vs. RoBERTa (general-purpose) tests whether financial pre-training provides a measurable advantage.
- Robust training: Class weighting, early stopping, learning rate scheduling, and gradient clipping ensure stable training on a small, imbalanced dataset.
- Comprehensive evaluation: Macro F1, per-class metrics, confusion matrices, and error analysis provide deep insight into model behavior.
- Multilingual extension: XLM-RoBERTa demonstrates cross-lingual transfer to Spanish and Russian.
- Production-ready: SentimentPredictor provides a clean API for real-world deployment.