Financial Sentiment Analysis

Project Overview

Selivankin Kirill
Group: BDA-2304

Three-class sentiment classification of financial texts
using transformer models (FinBERT, RoBERTa, XLM-RoBERTa)
and LLM evaluation (Qwen2.5-3B)
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? The project also explores data augmentation for class balancing, multilingual transfer learning (XLM-RoBERTa), and evaluates a local LLM (Qwen2.5-3B) as a zero-shot baseline.

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, augmentation, LLM)
data/raw/            - Original dataset
data/processed/      - Cleaned dataset
data/splits/         - Train/val/test CSVs
data/augmented/      - Balanced training set
outputs/models/      - Saved model checkpoints
outputs/figures/     - Generated plots
outputs/reports/     - Statistics and evaluation reports
experiments/         - Results JSON with all metrics
```

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. Data Augmentation

The original training set is heavily imbalanced (negative: 294, neutral: 1,502, positive: 621). To improve model performance on minority classes, we balance the training set using three data sources in priority order. Only the training set is augmented. Validation and test sets remain untouched to ensure honest evaluation on the natural data distribution.

Augmentation strategy

Source	Description	Samples Added
PhraseBank 50agree	Additional sentences from the 50% agreement subset	1,052
nlpaug (delete)	Random word deletion applied to minority-class sentenc	168
Templates	50 financial templates per class with slot-filling	869

Balanced training set

Class	Before	After	Source Breakdown
Negative	294	1,502	294 orig + 310 phrasebank + 168 aug + 730 template
Neutral	1,502	1,502	1,502 original (no augmentation needed)
Positive	621	1,502	621 orig + 742 phrasebank + 139 template
Total	2,417	4,506	

The augmented dataset is saved to data/augmented/train_balanced.csv with columns: sentence, label, label_name, source (original, phrasebank_50agree, augmented_delete, template_generated).

6. 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	278M	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.

7. 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).

8. 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 all models.

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.

9. Results

Fine-tuned model performance (English, balanced data)

Model	Accuracy	F1 (weighted)	F1 (macro)	Best Epoch
FinBERT	95.56%	0.956	0.950	2
RoBERTa	91.12%	0.914	0.909	2
XLM-RoBERTa	91.70%	0.918	0.895	--

Key Finding: FinBERT outperforms RoBERTa by 4.44% in accuracy, confirming that domain-specific pretraining on financial corpora provides meaningful improvements for financial sentiment analysis. Both hypotheses are confirmed: H1 (macro F1 > 0.85) and H2 (FinBERT > RoBERTa by > 3%).

Base models vs fine-tuned (baseline comparison)

To quantify the value of fine-tuning, we evaluate pre-trained models without any task-specific training:

Model	Accuracy	F1 (weighted)	F1 (macro)
RoBERTa (zero-shot)	39.96%	0.273	0.415
Qwen2.5-3B (zero-shot)	81.85%	0.800	0.781
RoBERTa (fine-tuned)	91.12%	0.914	0.909
XLM-RoBERTa (fine-tuned)	91.70%	0.918	0.895
FinBERT (base)	94.79%	0.949	0.935
FinBERT (fine-tuned)	95.56%	0.956	0.950

Fine-tuning improvement

Model	Accuracy Gain	F1 (weighted) Gain
FinBERT	+0.77%	+0.008
RoBERTa	+51.16%	+0.641

FinBERT's pre-trained sentiment head already achieves 94.8% accuracy on financial texts - fine-tuning on the target dataset adds only marginal improvement (+0.8%). RoBERTa, lacking any financial sentiment knowledge, jumps from 40% to 91% with fine-tuning (+128% relative gain).

10. LLM Evaluation (Qwen2.5-3B)

To assess how a general-purpose local LLM compares to fine-tuned transformers, we evaluate Qwen2.5-3B (running via Ollama) on the same 518-sample test set using zero-shot prompting.

Model	Accuracy	F1 (weighted)	F1 (macro)
RoBERTa (zero-shot)	39.96%	0.273	0.415
Qwen2.5-3B (zero-shot)	81.85%	0.800	0.781
RoBERTa (fine-tuned)	91.12%	0.914	0.909
FinBERT (base)	94.79%	0.949	0.935
FinBERT (fine-tuned)	95.56%	0.956	0.950

Qwen2.5-3B achieves 81.85% accuracy with zero-shot prompting - significantly better than RoBERTa zero-shot (39.96%) but well below fine-tuned models. The main weakness is positive class recall (44%): the LLM tends to classify positive sentences as neutral. Out of 94 errors, 73 (77.7%) are positive-to-neutral misclassifications.

This suggests the model treats subtle financial optimism (e.g. 'operating profit rose to EUR 17.5 mn') as factual rather than positive. Inference time: ~22 min for 518 samples (2.5s/sample) vs seconds for fine-tuned models.

LLM per-class F1

Class	Qwen2.5-3B	FinBERT (fine-tuned)	Gap
Negative	0.882	0.953	-0.071
Neutral	0.871	0.968	-0.097
Positive	0.589	0.930	-0.341

11. Cross-lingual Results (XLM-RoBERTa)

XLM-RoBERTa enables zero-shot cross-lingual transfer - the model is trained on English data and tested on Spanish without any Spanish training examples.

Language	Accuracy	Notes
English (test set)	91.7%	Standard evaluation
Spanish (zero-shot)	80.0%	No Spanish training data
Transfer Efficiency	87.2%	Spanish / English ratio

Spanish per-class performance

Class	Accuracy	Correct/Total
Positive	100%	5/5
Neutral	100%	5/5

Negative	40%	2/5

The model struggles with negative Spanish sentences, likely because negative financial expressions differ more across languages than positive/neutral ones.

12. 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}
```

13. 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 all models.

14. Notebooks

01_data_analysis.ipynb - Exploratory Data Analysis

This notebook runs the full EDA pipeline:

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

02_model_training.ipynb - Training and Evaluation

This notebook trains and evaluates both models:

- Creates stratified train/val/test splits.
- Loads balanced training set (toggle USE_BALANCED = True/False).
- Trains RoBERTa (general baseline) and FinBERT (domain-specific) with early stopping.
- Evaluates both models on the test set with classification reports and confusion matrices.
- Performs error analysis and model comparison.
- 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).

03a_data_augmentation.ipynb - Dataset Balancing

This notebook balances the training set from 2,417 to 4,506 samples using three sources: additional PhraseBank sentences (50% agreement), nlpaug word deletion, and template-generated sentences.

03b_baseline_comparison.ipynb - Base vs Fine-tuned

This notebook evaluates pre-trained (non-fine-tuned) models on the same test set: FinBERT base with its pre-trained sentiment head, and RoBERTa via zero-shot NLI classification.

04_llm_evaluation.ipynb - LLM Evaluation

This notebook evaluates Qwen2.5-3B (via Ollama) on the same test set using zero-shot prompting, comparing LLM performance against all fine-tuned transformer models.

15. 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)
    |
    v
augmentor.py  -->  Balanced training set (4,506 samples)
    |               (PhraseBank 50agree + nlpaug + templates)
    v
classifier.py -->  SentimentClassifier (FinBERT / RoBERTa / XLM-R)
    |
    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}
```

16. Summary

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

- Modular architecture: Each component (data, models, visualization) is independent and reusable.
- Five-model comparison: FinBERT (domain-specific) vs. RoBERTa (general-purpose) vs. XLM-RoBERTa (multilingual) vs. base/zero-shot baselines vs. LLM (Qwen2.5-3B).
- Data augmentation: Balances the training set from 2,417 to 4,506 samples using PhraseBank 50agree, nlpaug, and template generation.
- 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, error analysis, baseline comparison, and LLM evaluation provide deep insight into model behavior.

- Multilingual extension: XLM-RoBERTa demonstrates cross-lingual transfer to Spanish (80% accuracy).
- LLM comparison: Qwen2.5-3B zero-shot (81.85%) shows LLMs are useful but significantly trail fine-tuned domain-specific models (FinBERT 95.56%).
- Production-ready: SentimentPredictor provides a clean API for real-world deployment.