Financial Text Inference for Direction & Evaluation

Sanjna Suresh sanjnasuresh@umass.edu
Hrudayaditya Jallu hjallu@umass.edu
Pranav Reddy Pesaladinne ppesaladinne@umass.edu

1 Problem statement

[1]

In today's fast-paced financial markets, stock price prediction has been a key area of focus for investors, traders, and researchers. Stock price movements are known to have a significant impact on the overall economy. These prices are highly dynamic and influenced by numerous factors such as operations, market conditions, and policies. When considered collectively, these factors make stock data complex, volatile, and nonlinear. Consequently, inaccurate predictions can lead to substantial losses—not only for companies and their investors but also for the broader national economy. Therefore, improving the accuracy of stock price prediction is crucial, not just for corporate and investor gain but for the economic welfare of the country as a whole.

In recent years, traditional and statistical stock prediction methods have evolved with the integration of machine learning and deep learning techniques for forecasting. Conventional approaches such as ARIMA and GARCH have been widely used for this task, relying primarily on numerical time-series data such as historical prices, trading volumes, and technical indicators. Subsequently, Recurrent Neural Networks (RNNs) were developed to model temporal dependencies and capture complex relationships in financial data, improving market prediction to a certain extent. However, RNNs suffered from major limitations, including difficulty in learning long-term dependencies and issues such as vanishing and exploding gradients. To overcome these drawbacks, Long Short-Term Memory (LSTM) networks were introduced. (Chung and Shin, 2018), in their work "Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction" (Sustainability, Vol. 10, No. 10), developed a hybrid model to predict the Korean stock exchange using genetically optimized LSTMs. Although their model outperformed traditional benchmarks, it still faced challenges due to the high computational cost and time required for learning both short- and long-term dependencies. Thus, while LSTMs marked a significant improvement, they remain insufficient for efficient large-scale financial forecasting. Hence, we aim to focus on additional factors that could improve the accuracy of stock price prediction. Previous research suggests that traditional models may not fully capture erratic human behavior and psychological influences that affect market sentiment—often driven by public perception of a company at a given point in time.

Recent studies reinforce this connection between news sentiment and market behavior. For instance, (Naeem et al., 2020) in "An Intelligent Stock Prediction System Using Sentiment Analysis on News Data" (Procedia Computer Science, Vol. 167) demonstrated that textual information extracted from financial news articles can significantly improve prediction accuracy when combined with numerical data. Similarly, (Xie et al., 2018) in "Stock Market Prediction Based on Text Mining Technology: A Support Vector Machine Approach" (Frontiers in Psychology, Vol. 9, Article 1306) highlighted a strong interdependency between public news sentiment and stock price fluctuations, emphasizing the critical role of textual signals in forecasting models.

This project aims to leverage news-based information for enhanced stock forecasting by evaluating multiple deep learning architectures, including BERT, FinBERT, and DistilBERT, alongside various pruning and model compression strategies. Through this framework, we seek to analyze how different architectures and pruning tech-

[1]The code for this project is available at: https://github.com/hrudayaditya/Financial-Text-Inference-for-Direction-Evaluation

niques influence predictive performance, computational efficiency, and model scalability. Additionally, we aim to determine whether lighter, pruned models can achieve comparable accuracy to their larger counterparts while significantly reducing inference time and resource usage.

Representative prior work includes (Chen et al., 2019), (Bhardwaj et al., 2015), and (Kalyani et al., 2016).

Latency has a direct impact on the quality of decisions made in financial trading systems. For real-time financial news streaming predictive models to be useful, their outputs must be produced in milliseconds. Even while large transformer models like BERT and FinBERT are accurate, their ongoing deployment is expensive and computationally demanding. Additionally, model size is a significant cost consideration because stock market prediction models are frequently retrained every day on enormous amounts of data. It is computationally costly and unfeasible for smaller financial organisations or academic research sets to run big transformer models like FinBERT on each fresh batch of financial news. In order to achieve comparable predictive accuracy with significantly reduced inference time and resource costs in terms of computation and storage, as well as improved accessibility and scalability that allow near-real-time integration into trading and monitoring systems without the performance trade-off, we investigate model pruning and compression.

In addition to increasing efficiency, it's crucial to make sure these models persist to be accurate and acceptable in a variety of financial scenarios. Market behaviour varies greatly between industries and changes with more general economic regimes. For eg. A model that performs well in technology stocks during stable periods may behave unpredictably in the energy sector or during market volatility. Using ticker to industry mappings and indicators like the SandP 500 and VIX indices, our study will examine model performance by sector and market environment in order to solve this. In order to facilitate modifications that make prediction models not only faster but also more reliable and trustworthy for real world deployment, this deeper examination seeks to identify how and when prediction models fail.

2 What you proposed vs. what you accomplished

- Collect and preprocess a financial news dataset with entity-level sentiment annotations. **Completed**. We used a human-labeled financial news dataset and converted entity-level sentiment annotations into a supervised classification format.

- Build and evaluate classical baseline models for financial sentiment analysis. **Completed**. We implemented TF-IDF based models with linear classifiers to establish strong non-neural baselines.

- Train transformer-based models for financial sentiment classification. **Completed**. We fine-tuned FinBERT and DeBERTa-v3 models and evaluated them using accuracy and macro-F1 on held-out data.

- Improve performance beyond baseline models using more advanced architectures. **Completed**. Both FinBERT and DeBERTa achieved substantial improvements over classical baselines, with DeBERTa yielding the strongest overall performance.

- Analyze model robustness and failure modes through detailed error analysis. **Completed**. We performed a manual and automated error analysis on misclassified test examples to identify common semantic and syntactic patterns associated with model errors.

- Evaluate pruning and model compression techniques to study efficiency tradeoffs. **Not completed**. Due to time constraints, we prioritized model comparison and error analysis over efficiency-oriented experiments.

3 Related work

Stock market forecasting has been explored through a variety of approaches ranging from traditional statistical methods to modern machine learning techniques. Lu Chen et al. (2019) (Chen et al., 2019) proposed a hybrid attention-based EMD–LSTM model for financial time-series prediction (2019 2nd International Conference on Artificial Intelligence and Big Data, IEEE), which utilized historical stock data to capture temporal

dependencies and improve forecast accuracy. Although this kind of method shows that deep sequence models can outperform traditional statistical baselines on volatile financial signals, it still mainly relies on numerical history and does not directly model the impact of real-time information (such as breaking news) that can result in abrupt shifts. **In contrast, our goal is not only to model temporal dynamics from numeric history but to incorporate news-driven signals and explicitly study the accuracy–efficiency tradeoff under real-time inference constraints.**

Sentiment, on the other hand, captures psychological and behavioral factors that are absent from price histories. (Bhardwaj et al., 2015) emphasized the influence of sentiment analysis on stock prediction, demonstrating that investor sentiment derived from Sensex and Nifty data plays a crucial role in determining stock market trends. In recent years, researchers have increasingly turned to news sentiment analysis as an important predictor of stock movement. J. Kalyani et al. (2016) (Kalyani et al., 2016) explored this relationship in *Stock Trend Prediction Using News Sentiment Analysis* (IJCSIT, Vol. 8, No. 3), showing that textual data from financial news can effectively complement numerical indicators. Similarly, A. Giuseppe et al. (2019) (Giuseppe et al., 2019) in *Combining News Sentiment and Technical Analysis to Predict Stock Trend Reversal* (ICDMW 2019) integrated technical indicators with sentiment information to enhance predictive performance.These studies encourage the use of text as a primary signal, particularly in situations where prices react rapidly to new information. **Although these studies show the value of text, they typically do not address the question of whether transformer-based text models can be made fast and affordable enough for millisecond-level deployment. Our work directly addresses this question through pruning/compression and latency-aware evaluation.**

As datasets and compute grew, researchers moved from hand-engineered sentiment features to deep neural architectures that learn representations from text. Extending this line of research, K. Zeynep et al. (2019) (Kılıç et al., 2019) compared several deep learning architectures—BERT, LSTM, RNN, and CNN—for predicting stock direction in the Turkish market, with BERT achieving the highest accuracy of 96.27%. Likewise, Matheus Gomes Sousa et al. (2019) (Sousa et al., 2019) compared CNN and pre-trained BERT models for financial sentiment analysis, concluding that BERT consistently outperformed CNN in both accuracy and contextual understanding. (Hu et al., 2018) proposed *Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction*, which used deep neural architectures to capture hidden patterns between financial news and stock price fluctuations, outperforming several traditional methods and (Sun et al., 2014) developed sentiment classification techniques using Word2Vec for the Hong Kong stock exchange. Their findings established that transformer-based models such as BERT significantly outperform older embedding methods like FastText in handling semantic nuances of financial text. **Although these works imply that text is a valuable input, they hardly ever inquire if transformer-based text models can still function at a millisecond-level deployment but our research is focused on that question most directly by pruning/compression and latency-aware evaluation.**

Among the above, the most relevant to our setting are (1) transformer-based comparisons for market prediction and financial sentiment (Kılıç et al., 2019; Sousa et al., 2019) and (2) news-oriented deep prediction frameworks (Hu et al., 2018). The deployment constraint that is essential to real-time trading systems—end-to-end latency and scalability under frequent retraining—is not fully addressed in these papers, which mainly concentrate on predictive accuracy. Large transformer models can be expensive to deploy repeatedly in practice, and inference frequently needs to happen in milliseconds. **What distinguishes our approach is that we (i) compare BERT-related backbones with specific financial and lightweight models, and (ii) view inference latency and scalability as essential objectives equally to those of accuracy.**

Collectively, these studies highlight the growing dominance of transformer-based models such as BERT, FinBERT, and DistilBERT in financial forecasting tasks. However, the computational demands of these large models pose challenges for scalability and real-time deployment—motivating the need to explore model pruning and compression techniques for efficient, high-performance stock prediction systems. **Our work is to systematically examine the trade-off of accuracy vs. efficiency for BERT-family models with prun-**

ing/compression and to investigate the locations where the model performance is kept or lost in different sector and regime slices to be able to deploy the model in a realistic way.

4 Your dataset

Dataset source: We use **SEntFiN** (Sentiment and Entity annotated Financial News), a human-annotated dataset of financial and economic news headlines sampled from Indian business news providers like *The Economic Times* over the years **2011–2015** (Sinha et al., 2022). The dataset is annotated at the **entity level**: each headline contains one or more recognized financial entities (e.g., a company, sector, or organization), and each entity is assigned one of three sentiment labels: `positive`, `negative`, or `neutral` (Sinha et al., 2022).

Task definition: Our task is **entity-aware sentiment classification on headlines**. Each training example is:

Input: (headline text, target entity)
→ Output: sentiment label in {pos, neg, neutral}.

This framing matches the dataset structure, where a single headline may contain multiple entities and thus multiple sentiment labels.

Dataset description: SEntFiN contains **10,753** headlines and approximately **14.4K** total entity-sentiment annotations. The class distribution is fairly balanced: about **35%** positive, **26%** negative, and **38%** neutral. Headlines are short on average (roughly **10 words**), but multi-entity headlines are slightly longer.

A key property is that a substantial subset of headlines contains multiple entities, and many of those contain **conflicting sentiments** across entities (e.g., one entity positive while another negative). This is important for our modeling because the same words must be interpreted differently depending on which entity is the target.

Dataset Challenges:

- **Multiple entities per headline:** one headline can mention several entities with different sentiment labels, so models must condition sentiment on the *target entity* rather than produce one headline-level polarity.
- **Conflicting sentiments:** mixed sentiment is common in multi-entity headlines, which confuses models that rely on overall headline polarity.
- **Entity surface-form variation:** companies and sectors appear under multiple aliases/phrases; the dataset work discusses an entity database to support consistent entity handling.

Example input/output pairs. Below are representative examples of the task format:

- **Headline:** *"Negative on Chambal, Advanta: Mitesh Thacker"*
  Target entity: Chambal → **Label:** negative.
  Target entity: Advanta → **Label:** negative.

Note on dataset version. The paper reports the above statistics for the SEntFiN release v1 it describes (Sinha et al., 2022).

4.1 Exploratory data analysis (EDA)

Before training, we performed a lightweight exploratory analysis to understand (i) the class distribution and (ii) the typical headline length.

Sentiment label distribution. Figure 1 shows the sentiment label counts in SEntFiN. The dataset is not perfectly balanced: **neutral** is the most frequent class, followed by **positive**, and **negative** is the least frequent. This motivates reporting metrics beyond accuracy (e.g., Macro-F1) and using stratified splits to preserve class proportions across train/validation/test.

Headline length distribution. Figure 2 shows the distribution of words per headline. Most examples are short (roughly between **7–13 words**), with a long tail of longer headlines. This supports our preprocessing choices (short max sequence length and truncation), and it also helps explain why classification can be challenging: short headlines often provide limited context, which can increase ambiguity between **neutral** and weakly **positive/negative** statements.

4.2 Data preprocessing

Our raw dataset is provided as a CSV where each row contains a financial news headline and a dictionary-like field mapping one or more entities to their sentiment labels. We preprocess this data by converting it into a supervised learning table suitable for model fine-tuning.

First, we parse the entity-to-sentiment field (stored as a string) into a Python dictionary. Since

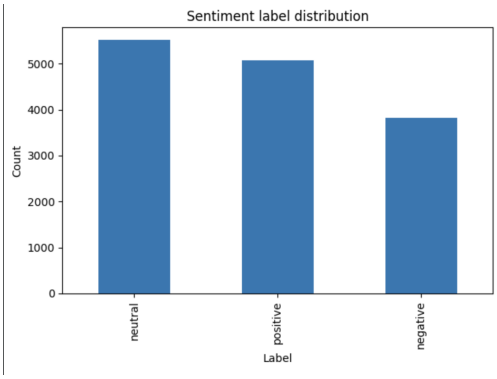

Figure 1: Sentiment label distribution in SEntFiN (entity-expanded instances). Neutral is the most frequent class, followed by positive and then negative.

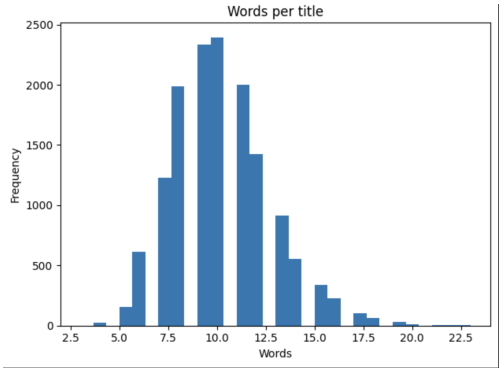

Figure 2: Words per title in SEntFiN. Headlines are short on average, with most falling around 7–13 words and a smaller tail of longer titles.

a single headline can contain multiple entities, we then expand the dataset into an entity-level format by creating one training instance per entity. Each resulting row contains the original headline text, a single entity, and that entity's sentiment label. Next, we map the sentiment labels into integer class IDs (three classes: negative, neutral, positive) to match the classification head used during fine-tuning.

Finally, we split the resulting labeled instances into train/validation/test partitions using stratified sampling to maintain approximately the same label distribution across splits. For model input, we tokenize the headline text using the FinBERT tokenizer with truncation (fixed maximum sequence length) and batch padding handled during training.

4.3 Data annotation

Our project has no requirement to perform any new data annotation. It uses the sentiment labels already present in the dataset (entity → sentiment) as ground-truth targets for supervised training and evaluation. The preprocessing step described above simply restructures these provided labels into a format compatible with model fine-tuning.

5 Baselines

5.1 Baseline model: FinBERT fine-tuning (3-way sentiment)

Our baseline is a transformer-based sentiment classifier initialized from the pretrained checkpoint `ProsusAI/finbert`. We fine-tune `AutoModelForSequenceClassification` with `num_labels=3` to predict one of three sentiment classes: `negative`, `neutral`, and `positive`.

5.2 Baeline reasoning

We choose transformer-based baselines because our inputs are short financial news headlines and prior work shows contextual encoders are strong for finance sentiment. We include (1) a finance-domain BERT model (FinBERT) and (2) a finance sentiment model based on DeBERTa-v3. Simpler classical baselines (e.g., bag-of-words with linear classifiers) are omitted in the current implementation because our primary goal is to benchmark against strong pretrained finance-text encoders.

5.3 Dataset construction for baseline

The raw SEntFiN CSV contains a headline text field and an entity-to-sentiment mapping field (stored as a string). TWe first parse this mapping and then expands each headline into multiple supervised instances: one instance per entity in the mapping. Each resulting training row contains:

- **text**: the original headline string,
- **entity**: the entity name from the mapping,
- **label_str**: the sentiment label string (`negative`/`neutral`/`positive`),
- **label**: an integer ID used for training.

Sentiment labels are mapped to IDs using:

$$\texttt{negative} \to 0, \quad \texttt{neutral} \to 1, \quad \texttt{positive} \to 2.$$

5.4 Train/validation/test split

We perform a stratified split over the expanded instances (stratified by the integer sentiment label). We first split the dataset into 80% train and

20% temporary using `random_state=42`. The temporary 20% is then split evenly into validation and test sets (10%/10%), again stratified and using `random_state=42`. Concretely:

- Train: 80%
- Validation: 10%
- Test: 10%

Split granularity The split is performed **row-wise** after expanding text-data into multiple entity-level rows. As a result, the same text-data may appear in multiple splits if it generates multiple entity rows.

5.5 Baseline 1: FinBERT fine-tuning (3-way sentiment)

Model. We fine-tune a transformer classifier initialized from `ProsusAI/finbert` using `AutoModelForSequenceClassification` with `num_labels=3`.

Tokenization. We tokenize each headline using the FinBERT tokenizer with truncation to a maximum sequence length of 128 tokens. Padding is handled at batch time using `DataCollatorWithPadding`.

Training configuration. We fine-tune using HuggingFace `Trainer` with the following `TrainingArguments` values:

- `num_train_epochs = 3`
- `learning_rate = 2e-5`
- `per_device_train_batch_size = 16`
- `per_device_eval_batch_size = 32`
- `weight_decay = 0.01`
- `logging_steps = 50`
- `save_steps = 500`
- `save_total_limit = 2`
- `report_to = "none"`
- `load_best_model_at_end = False`

Evaluation. We evaluate using Accuracy and Macro-F1, and we additionally generate a test confusion matrix and a per-class classification report (precision/recall/F1).

5.6 Baseline 2: DeBERTa-v3 fine-tuning (3-way sentiment)

Model. Our second baseline fine-tunes a DeBERTa-v3 based sentiment model, initialized from the pretrained checkpoint **mrm8488/deberta-v3-ft-financial-news-sentiment-analysis**, using **AutoModelForSequenceClassification** with **num_labels=3**. This baseline provides a competitive comparison to FinBERT using a different transformer family.

Tokenization. We tokenize each headline using the tokenizer associated with the DeBERTa-v3 checkpoint. We use truncation with a maximum sequence length of 128 tokens and apply dynamic padding during batching via `DataCollatorWithPadding`.

Training configuration. We fine-tune the DeBERTa-v3 baseline using HuggingFace `Trainer`.

- `num_train_epochs = 3`
- `learning_rate = 2e - 5`
- `per_device_train_batch_size = 8`
- `per_device_eval_batch_size = 16`
- `weight_decay = 0.01`

We keep the same split and metrics as the FinBERT baseline for a fair comparison.

Evaluation. We evaluate DeBERTa-v3 using Accuracy and Macro-F1, and we also produce a confusion matrix and per-class metrics on the test set.

6 Your approach

Our approach focuses on entity-aware financial sentiment classification using transformer-based language models. Each example consists of a short news title paired with a sentiment label indicating whether the expressed sentiment toward a financial entity is positive, neutral, or negative.

We first construct a supervised dataset from a human-annotated financial news corpus and split it into training, validation, and test sets. As baselines, we implemented TF-IDF based models with linear classifiers. These baselines rely on surface-level lexical features and are expected to fail on semantically ambiguous or context-dependent headlines.

Our main models are transformer classifiers. We fine-tune FinBERT(FinBERT.ipynb) and DeBERTa-v3(DeBERTa-v3.ipynb) using supervised cross-entropy loss. These models are expected to better capture semantic structure but may still fail on mixed sentiment or multi-entity headlines. All training and evaluation code was implemented by us using the Hugging Face Transformers library and PyTorch. Pretrained checkpoints were used only for initialization.

Experiments were run on Google Colab with A100 GPU support. Due to memory constraints, we used small batch sizes and limited sequence lengths. No major Colab-specific workarounds were required beyond these configuration choices.

Both transformer models substantially outperformed the classical baselines. DeBERTa achieved the best overall performance, with a small but consistent improvement over FinBERT. These results indicate that transformer models are effective for financial sentiment classification, while remaining limited by dataset ambiguity rather than model capacity.

In addition to evaluating individual transformer models, we explored an ensemble approach that combines FinBERT and DeBERTa(FinBERT+DeBERTa-Ensemble.ipynb) predictions through probability averaging. The motivation was that the two models make partially different errors and could therefore complement each other. When both fine-tuned models were available in the same runtime, the ensemble is expected to produce a small improvement over the stronger single model on validation data.

However, this experiment required both trained models to be simultaneously loaded for inference. Due to storage and memory limitations even with Google Colab Pro, as well as exhausted compute credits, we were unable to consistently retain both fine-tuned models in the same environment. As a result, the ensemble experiment could not be run reliably across multiple repetitions. Hence we are treating the ensemble as an exploratory result rather than a primary contribution.

7 Results

We report results for two transformer baselines trained: **FinBERT** (`ProsusAI/finbert`) and **DeBERTa-v3** (**mrm8488/deberta-v3-ft-financial-news-sentiment-analysis**). We evaluate on the held-out test split using **Accuracy** and **Macro-F1**. For Accuracy, we also report a 95% binomial (Wilson) confidence interval.

Class	Precision	Recall	F1	Support
Negative	0.784	0.866	0.823	382
Neutral	0.805	0.763	0.783	552
Positive	0.823	0.805	0.814	507

Table 1: FinBERT test performance by class.

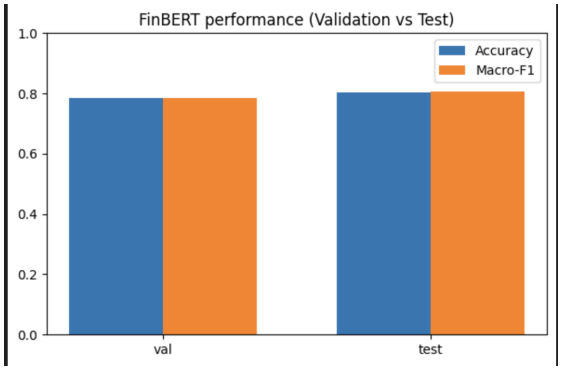

Figure 3: FinBERT validation vs. test performance (Accuracy and Macro-F1).

7.1 FinBERT

7.1.1 Overall performance

On the test set (N = 1441 instances), FinBERT achieves an accuracy of **0.805** (1160/1441 correct) and a macro-F1 of **0.807**. To quantify uncertainty for accuracy, we report a 95% binomial (Wilson) confidence interval:

$$\text{Accuracy} = 0.805, \quad 95\%\ \text{CI} \approx [0.784,\ 0.825].$$

7.1.2 Per-class performance

Table 1 reports per-class precision, recall, and F1 computed from the test confusion matrix. Performance is reasonably balanced across classes, with the lowest recall observed for the neutral class. We also report per-class Precision/Recall/F1 (Fig. 5), since overall accuracy can hide class-specific weaknesses.

7.1.3 Error patterns (from confusion matrix)

Figure 4 shows the test confusion matrix. We defer detailed analysis of common confusions and representative failure cases to Section 8.

7.2 DeBERTa-v3

7.2.1 Overall performance.

On the same test set (N=1441), DeBERTa-v3 achieves **Accuracy = 0.822** (1184/1441) and **Macro-F1 = 0.823**. The 95% Wilson confidence

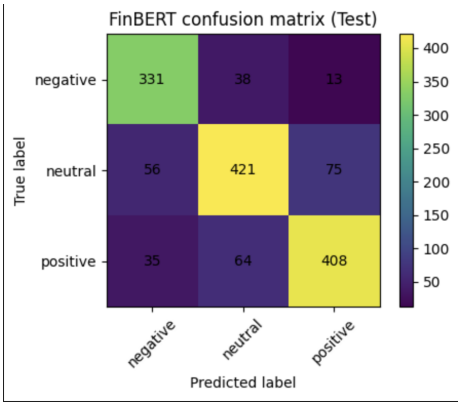

Figure 4: FinBERT confusion matrix on the test split.

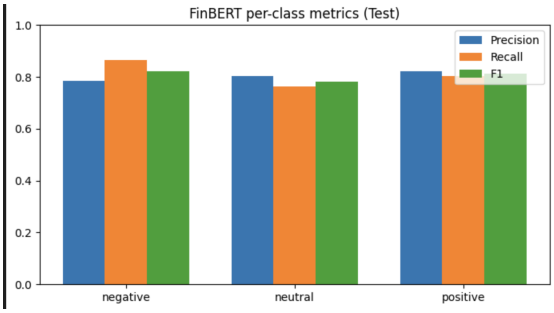

Figure 5: FinBERT per-class Precision, Recall, and F1 on the test set.

interval for accuracy is approximately **[0.801, 0.841]**.

7.2.2 Validation vs. test.

Figure 6 reports validation vs. test performance (Accuracy and Macro-F1).

7.2.3 Per-class performance.

Table 2 summarizes per-class Precision/Recall/F1 computed from the test confusion matrix.

7.2.4 Confusion matrix.

Figure 7 shows the DeBERTa-v3 confusion matrix on the test split.

8 Error analysis

8.1 FinBERT

8.1.1 Confusion-matrix driven error patterns

Figure 4 shows that most errors involve the **neutral** class. Out of **281** total test errors, **131** are cases where the true label is **neutral** but the model predicts **positive** or **negative** (56 neutral→negative and 75 neutral→positive). The single largest confusion pair is **neutral↔positive**: 75 neutral→positive and 64 positive→neutral (139 errors total). This indicates that the decision

Class	Precision	Recall	F1	Support
Negative	0.795	0.874	0.833	382
Neutral	0.835	0.761	0.796	552
Positive	0.830	0.848	0.839	507

Table 2: DeBERTa-v3 test performance by class.

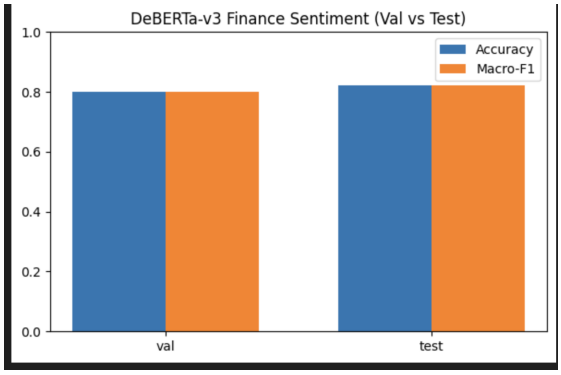

Figure 6: DeBERTa-v3 validation vs. test performance (Accuracy and Macro-F1).

boundary between neutral and positive is the hardest for the baseline, which is also visible in the per-class metrics (Fig. 5).

8.1.2 Class-specific weaknesses

Using the per-class metrics on the test set (Fig. 5):

- **Neutral has the lowest recall** (≈0.763). Many neutral examples are pushed into the positive/negative classes, suggesting that the model often interprets informational headlines as sentiment-bearing.
- **Negative has the highest recall** (≈0.866) but lower precision (≈0.784), meaning the model retrieves negative examples well but also produces some false negatives by mislabeling non-negative headlines as negative.
- **Positive is comparatively stable** (precision ≈0.823, recall ≈0.805), but it is still frequently confused with neutral (positive→neutral: 64 cases).

8.1.3 Error analysis via lexical properties

The model does a lightweight property-based error analysis in addition to aggregate metrics. Simple boolean indicators like has modal (modal verbs like may/could), has quotes (quoted speech), has negation, has uncertainty (hedging/uncertainty terms), punctuation cues (has question, has exclamation), and surface markers (has money, has url,

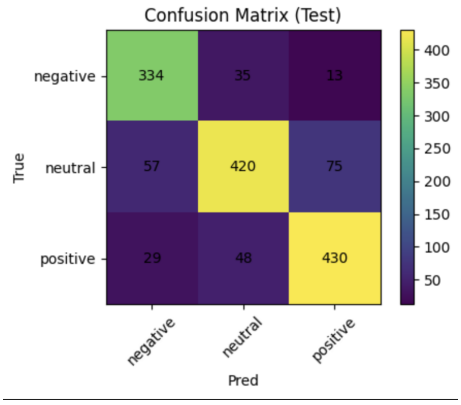

Figure 7: DeBERTa-v3 confusion matrix on the test split.

has ticker, and very long) are calculated for every test headline. Next, it compares the frequency of each attribute in samples that were incorrectly identified vs those that were successfully classified (Fig. 8). (Fig. 8).

Figure 8 shows demonstrates that quoted speech and modal/hedged language are significantly more prevalent in mistakes than in accurate predictions. In actuality, these structures (such as conjecture about future events, attribution to a speaker, or reported viewpoints) frequently weaken or indicate the sentiment signal, making it more difficult to deduce the polarity from a brief headline. Additionally, models are known to struggle with scope and reversal cues (e.g., not, no, never), which is consistent with the over representation of negation in errors compared to right situations. Additionally, we see modest increases for question markers and uncertainty, which again highlights ambiguous or noncommittal answers as difficult situations. In contrast, properties like `has_url`, `has_ticker`, and `very_long` appear rarely overall and do not meaningfully distinguish errors from correct predictions in this dataset.

8.1.4 Most common confusion pairs

To identify which labels are most frequently confused, we tabulate the most common $(y_{\text{true}}, y_{\text{pred}})$ pairs on the test set (Table 3). The dominant error mode is **neutral↔positive** confusion, followed by **neutral→negative**. Extreme flips such as **negative→positive** occur much less often.

8.1.5 Property-based error analysis

We developed a lightweight **property-based analysis** by computing simple boolean indicators for each headline (e.g., whether it contains modal/hedging language, quotes, negation, numeric cues, etc.). It then compares how often each property appears among **misclassified** vs. **correctly classified** examples. Figure 9 reports the fraction of examples in each group that contain the property, along with the difference Δ = error_rate − correct_rate.

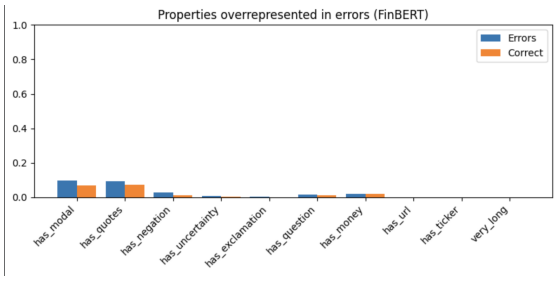

Figure 8: Properties overrepresented in errors (FinBERT). Bars show the fraction of misclassified vs. correctly classified test examples that contain each lexical/surface property.

True label	Predicted label	Count
neutral	positive	75
positive	neutral	64
neutral	negative	56
negative	neutral	38
positive	negative	35
negative	positive	13

Table 3: Most common FinBERT confusion pairs on the test set.

Properties with the largest positive deltas are **has_modal** ($\Delta = 0.0263$) and **has_quotes** ($\Delta = 0.0193$), indicating that hedged or attributed statements (e.g., speculation, reported speech) are more likely to be misclassified. **Negation** is also overrepresented in errors ($\Delta = 0.0164$), consistent with the known difficulty of handling polarity reversal cues in short texts. In contrast, numeric and directional cues such as **has_number**, **has_percent**, and **has_up_down** are *underrepresented* in errors (negative deltas), suggesting these explicit signals may make sentiment easier for the model to infer.

8.1.6 Why these errors likely happen (task- and pipeline-specific)

- **Weak sentiment cues and concise headlines.** Sentiment is implied rather than stated in financial headlines because they are brief and frequently factual. This explains the ob-

property	error_rate	correct_rate	delta_error_minus_correct
has_modal	0.096085	0.069828	0.026258
has_quotes	0.092527	0.073276	0.019251
has_negation	0.028470	0.012069	0.016401
has_uncertainty	0.007117	0.002586	0.004531
has_exclamation	0.003559	0.000000	0.003559
has_question	0.014235	0.012931	0.001304
has_money	0.021352	0.020690	0.000663
has_url	0.000000	0.000000	0.000000
has_ticker	0.000000	0.000000	0.000000
very_long	0.000000	0.000000	0.000000
has_allcaps_word	1.000000	1.000000	0.000000
very_short	0.028470	0.049138	-0.020668
has_percent	0.085409	0.117241	-0.031832
has_number	0.231317	0.312931	-0.081614
has_up_down	0.085409	0.176724	-0.091315

Figure 9: Property-based error analysis for FinBERT. "Error rate" and "Correct rate" are the fractions of misclassified vs. correctly classified test examples that contain each property.

served neutral-heavy confusions by making neutral vs. positive/negative intrinsically ambiguous.

- **Entity-level labels but text-only inputs.** A single headline is expanded into several entity-level instances during our preprocessing, but the baseline input consists solely of the headline text (the entity string is not injected into the model input). The same text is paired with different labels across instances when a headline contains multiple entities with different sentences. This can confuse a text-only classifier and promote neutral, positive, and negative mixing.

- **Borderline "good news" vs. "no sentiment" wording.** Many headlines describe events (e.g., results, plans, regulatory actions) where it is unclear whether the correct label should be neutral or positive without additional context. This aligns with the dominant neutral↔positive confusions.

8.2 DeBERTa-v3

8.2.1 Confusion-matrix driven error patterns

From the DeBERTa-v3 test confusion matrix (Fig. 7), there are **257** total errors out of 1441 test instances, fewer than FinBERT. Similar to FinBERT, most errors involve **neutral**:

- **Neutral→Negative/Positive:** 57 neutral→negative and 75 neutral→positive (**132** errors total).

- The largest bidirectional confusion pair is again **neutral↔positive**: 75 neutral→positive and 48 positive→neutral (**123** errors total).

Compared to FinBERT, DeBERTa-v3 reduces the positive→neutral direction (48 vs. 64), which contributes to the lower overall error count.

8.2.2 Class-specific weaknesses

DeBERTa-v3's per-class metrics (Table 2) indicate:

- **Neutral remains the hardest class** (recall ≈0.761), with many neutral examples still predicted as positive or negative.

- **Positive improves relative to FinBERT** (recall ≈0.848), consistent with fewer positive→neutral confusions.

- **Negative recall is also high** (≈0.874), and negative→positive confusions remain low (13 cases).

8.2.3 Why these errors likely happen (baseline-specific)

The DeBERTa-v3 uses the same overall pipeline structure as the FinBERT (entity-expanded labels, text-only inputs). Therefore, DeBERTa-v3 inherits similar failure modes.

9 Contributions of group members

- **Sanjna**: Led dataset understanding and preprocessing. Converted the raw financial news data into an entity-level supervised format and contributed to report writing, including dataset description and experimental setup. Also worked on the Ensemble of both the fine-tune models.

- **Hrudayaditya**: Designed and implemented the core modeling pipeline. Built and evaluated classical TF-IDF baselines, fine-tuned transformer models including FinBERT and DeBERTa, ran experiments on Google Colab, and analyzed performance tradeoffs. Also contributed to overall project direction and report writing.

- **Pranav**: Conducted detailed error analysis and manual inspection of misclassified examples. Annotated failure cases, identified common error patterns, and helped interpret model limitations. Contributed to the error analysis and discussion sections of the report. Also worked on the Ensemble of both the fine-tune models.

If you would like to privately share more information about the workload division that may have caused extenuating circumstances (e.g., a member of the group was unreachable and did no work), please send a detailed note to the instructors GMail account. We will take these notes into account when assigning individual grades.

10 Conclusion

In this project, we used the SEntFiN dataset to investigate entity-aware sentiment classification on financial news headlines and developed two robust transformer baselines: FinBERT and a DeBERTa-v3 financial sentiment model. Both models performed well, with DeBERTa-v3 slightly outperforming FinBERT on Macro-F1 and overall test Accuracy. However, our error analysis revealed that the dominant failure mode for both models is still separating *neutral* from *positive/negative* in brief, information-dense headlines. The project expands each headline into multiple entity-labeled instances, but our current baselines are text-only and do not explicitly condition on the target entity, which probably contributes to ambiguity in multi-entity and weak-sentiment cases. This was the most surprisingly challenging aspect of handling the dataset's entity-level supervision.

11 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Yes, we used ChatGPT, Gemini in google search, Gemini in colab, perplexity.

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - "Can you help me rephrase this section in better language? (Referring to Approach section)"
  - "–Error Message Here–, can you help me fix this!
  - "–Package Conflicts Here–", How do I fix this in colab!?
  - "I wanna do an error analysis on my FinBERT fine-tuned model, Suggest me some good approaches"
  - "If I actually ensemble DeBERTa-v3 and FinBERT fine-tuned model and try to build a new model, how would that be?"
  - "–Screenshots of colab behavior–" what's going on here!"
  - "Can I push my code to github directly from colab, like vs code style?"
  - "why is my ensemble model underperforming and how should I correctly ensemble FinBERT and DeBERTa?"
  - "How do I populate the fine tuned models in the ensemble model"
  - "–content– Is there a better way to format this in LaTeX?"
  - "–content– Is this good enough for a graduate level report?"
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
  - The AI was only used to get some suggesstions on what approaches we can try, support with fixing errors, and alot of cloab specific issues while running the code, we gave sentences and asked it give ideas for structure while doing the final report, we already had the main content written. We edited the AI's suggestions a bit. The AI's output was generally correct but while seeking help with fixing errors we were stuck for hours even with LLMs support, with the

report we still needed some rewriting and simplification to fit the final report.

References

Aditya Bhardwaj, Yogendra Narayan, Maitreyee Dutta, and et al. 2015. Sentiment analysis for indian stock market prediction using sensex and nifty. *Procedia Computer Science*, 70:85–91.

Lu Chen, Yonggang Chi, Yingying Guan, and Jialin Fan. 2019. A hybrid attention-based EMD-LSTM model for financial time series prediction. In *2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 113–118. IEEE.

H. Chung and K. S. Shin. 2018. Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability*, 10(10).

A. Giuseppe, C. Luca, G. Paolo, and B. Elena. 2019. Combining news sentiment and technical analysis to predict stock trend reversal. In *2019 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE.

Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 261–269.

J. Kalyani, N. Bharathi, and Jyothi Rao. 2016. Stock trend prediction using news sentiment analysis. *International Journal of Computer Science & Information Technology (IJCSIT)*, 8(3).

Zeynep Kılıç, Derya
"Ozge, and Mitat Uğur. 2019. Financial sentiment analysis for predicting direction of stocks using BERT and deep learning models. In *International Conference on Innovative Technologies (ICIIT-19)*.

Muhammad Naeem and 1 others. 2020. An intelligent stock prediction system using sentiment analysis on news data. *Procedia Computer Science*, 167:1110–1119.

Ankur Sinha, Satishwar Kedas, Rishu Kumar, and Pekka Malo. 2022. Sentfin 1.0: Entity-aware sentiment analysis for financial news. *Journal of the Association for Information Science and Technology*, 73(9):1314–1335.

Matheus Gomes Sousa, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes, and Edson Takashi Matsubara. 2019. BERT for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1597–1601. IEEE.

Zengcai Sun, Hua Xu, Dongwen Zhang, and Yunfeng Xu. 2014. Chinese sentiment classification using a neural network tool — word2vec. In *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*.

Jialin Xie, Yong Zhang, and 1 others. 2018. Stock market prediction based on text mining technology: A support vector machine approach. *Frontiers in Psychology*, 9:1306.

[2]

[2]The code for this project is available at: https://github.com/hrudayaditya/Financial-Text-Inference-for-Direction-Evaluation