15.776 Intensive Hands-on Deep Learning Final Project

Group 3: Shiqi Hong, Xiaoyan Wang, Alice Jiang, Kejing Yu

Link to our colab: HODL_Final Project_1207.ipynb

I. Problem Description

In the fast-moving world of financial markets, information is an important driver of asset prices. News headlines, in particular, contain qualitative signals that can impact investor sentiment and, consequently, stock returns in the short term. The core challenge lies in transforming this unstructured textual data into systematic, actionable trading signals. Manually processing the large volume of financial news in real time is impractical, which places strong demands on automated sentiment models to accurately extract nuanced, market-relevant information.

This project investigates whether the sentiment embedded in company-specific news can predict near-term stock movements for a universe of large-cap technology stocks. To that end, we first develop and compare multiple sentiment classification models, including a rule-based approach (VADER), a pre-trained financial transformer (FinBERT), and a classical machine learning model (SVM), before combining them into a final ensemble classifier that labels headlines as Positive, Neutral, or Negative. We then use these labels to build a daily-rebalanced, long-only portfolio that takes positions following positive news and evaluates its out-of-sample performance against the S&P 500 benchmark.

II. Approach

1. Creating Our Own Dataset

Step 1: News Data

We collected firm-level news headlines from Reuters using the Refinitiv Data Library for a predefined universe of 200 large-cap technology companies, identified by their Reuters Instrument Codes (e.g., NVDA.O, AAPL.O, MSFT.O). Headlines were retrieved over the most recent 15-month period (approximately 450 days) to ensure sufficient coverage for sentiment modeling and strategy backtesting. To control query size and API usage, we capped the number of headlines at 1,000 per firm and restricted the data to English-language Reuters sources. All retrieved headlines were combined into a single dataset and exported to a CSV file for subsequent preprocessing and analysis.

Step 2: Price Data

We retrieved the historical adjusted close prices for all tickers using the Yahoo Finance API, ensuring that returns were computed on prices already adjusted for splits and dividends.

2. Model and Implementation

Step 1: Data Preprocessing

- ❖ **Overview:** 78,301 news headlines of 200 firms over a 15-month period are collected.
- ❖ **Data preprocessing steps taken:** time conversion, ticker extraction, company name mapping, cleaning of headlines, and creation of a new column that combines CompanyName and headline as the input for the models that we will implement later.

Step 2: Creation of Training, Validation, and Test Dataframe

- ❖ **Avoiding look-ahead bias:** we aim at leveraging our sentiment analysis model results to create stock-trading strategies in Part 3, so when training sentiment analysis models, we avoid look-ahead bias by splitting training, validation, and test data strictly according to their chronological orders.
- ❖ **Strategy_date Selection:** to ensure we have enough data for strategy building, we set a strategy_date = '2024/11/01' and only select training, validation, and testing data before that date. Correspondingly 7,448 data is separated for the sentiment analysis model training, fine tuning, and evaluation.
- ❖ **Dataframe Size:** 1,000 for training and validation data, 200 for test data.
- ❖ **Dataframe Creation and Sentiment Label Generation:** we identify a cutoff date that splits the 7,448 data evenly to 6 parts, and randomly select 1,000 data from the first 5 parts as train_val_df (the first 800 as train_df, the last 200 as val_df), and randomly select 200 data from the last part as test_df. We then send the data to ChatGPT and make it return one of the sentiment labels (Positive/Neutral/Negative).
  - ➢ Prompt: *You are an expert in financial news sentiment analysis. The text contains a company's name followed by a news headline. For each headline, classify the sentiment specifically from the perspective of the stock market or investor sentiment. Choose ONLY ONE label: Negative (bearish or adverse impact on the stock), Neutral (no clear impact on the stock), Positive (bullish or favorable impact on the stock). Return your output in JSON: {"input": "<input_text>", "sentiment": "Positive"}*
- ❖ **Backup Benchmark Dataframe:** SEntFiN database (a human-annotated dataset of news headlines with entity-sentiment annotations) used as an alternative benchmark.

Step 3: Preliminary Examination of Different Models' Performance

We use labeled_train_df for training if needed, and evaluate model performance on labeled_val_df (GPT-labeled val_df) and benchmark_df (SEntFiN).

- ❖ **Rule-based Model Vader:** we import it for direct usage and evaluate its performance. The accuracy on GPT-generated labels is 0.485, on SEntFiN is 0.53, this accuracy difference may indicate that GPT responses don't share the same patterns as SEntFiN.
- ❖ **Pretrained Financial Sentiment Model FinBert:** we tried out three different implementations. The best performing one is yiyanghkust/finbert-tone - Disable fine tuning. Fine tuning may reduce model performance because the GPT-labeled dataset is likely too small and noisy, causing the model to overfit and overwrite useful pre-trained knowledge.

    - ProsusAI/finbert - Disable fine tuning: the accuracy on GPT-generated labels is 0.25, on SEntFiN is 0.182.
    - yiyanghkust/finbert-tone - Disable fine tuning: the accuracy on GPT-generated labels is 0.55, on SEntFiN is 0.712.
    - yiyanghkust/finbert-tone - Enable fine tuning: we tried out different numbers of training epochs, and the parameters that we allow for finetuning. As the number of training epochs increase or as we allow for more parameters to be tuned, the model's performance decreases. Eventually, we use 1 training epoch, and only allow fine tuning for the classifier head. The corresponding accuracy on GPT-generated labels is 0.515, on SEntFiN is 0.678.
- **Support Vector Machine Model (SVM):** after model training, the accuracy on GPT-generated labels is 0.645, on SEntFiN is 0.494. This difference also indicates the different patterns between GPT responses and SEtFiN.

Step 4: Examination of Different Ensemble Models' Performance

Based on model performance listed on Step 3, we decided to try out three different methods to ensemble Vader, yiyanghkust/finbert-tone - Disable fine tuning, and the SVM model.

For this step, if the ensemble model doesn't need training (Majority Vote model), then we use train_val_df to train SVM model, and then generate sentiment labels on test_df. If the ensemble model needs training (Weighted Vote and Stacked Meta Model Vote), then we use train_df to train SVM model, use val_df to finetune parameters for ensemble model, and then generate sentiment labels on test_df.

- **Majority vote:** final vote is determined by the majority vote of the three sentiment analysis models. The accuracy on the GPT-generated labels is 0.65.
- **Weighted vote:** after training the SVM with train_df, we used the accuracy score of the three sentiment analysis models on val_df as weights. Their weighted average sentiment scores were then used to determine the final vote. The accuracy on the GPT-generated labels is 0.64.
- **Stacked Meta Model Vote:** we first combine train_df and val_df into a single training set, and use a TimeSeriesSplit to generate multiple chronological folds. For each fold, we train the base sentiment analysis models on the earlier portion and generate the out-of-fold predictions on the validation portion. These out-of-fold predictions form the meta-features used to train the meta-classifier. After the final meta-model is trained on all out-of-fold predictions, we train the base sentiment analysis models on train_df and val_df, and apply it to the test set to get test meta-features, and let the meta-model produce the final stacked ensemble predictions. The accuracy on the GPT-generated labels is 0.635.

Step 5: Final Model and Sentiment Labels Generation

Given the results on test data mentioned in Step 4, we choose Majority Vote as our final ensemble model. We then train the SVM model using a combination of train_df, val_df, and test_df, and then generate sentiment labels for all the data after the strategy_date using three sentiment analysis models (Vader, Finbert *(yiyanghkust/finbert-tone - Disable fine tuning)*, SVM). We then apply the Majority

Vote model to the labels generated by the three sentiment analysis models to get our final sentiment vote.

3. Trading Strategy

After obtaining the final sentiment votes for all news items, we constructed a daily-rebalanced trading strategy using the universe of 200 Tech stocks. The strategy is long-only: each day, we take long positions in stocks that received a positive sentiment signal on the previous day. If multiple news articles appear for the same stock on a given day, we apply a majority-vote rule to determine the final sentiment for the stock on that day. Positions are held for one day, consistent with the fact that the sentiment signal is derived from news published on that same one-day horizon.

We focus on a long-only implementation because our sentiment models are most reliable at identifying positive news sentiment, whereas signals for negative sentiment tend to introduce more noise and reduce performance.

III. Results

We compare our sentiment-based long-only strategy with the S&P 500 benchmark. As shown in the cumulative return plot, the strategy closely follows the market trend but delivers strong upside performance and faster recoveries after drawdowns.

The sentiment strategy achieves a Sharpe ratio of 1.36 during the testing period from 2024-11 to 2025-12, outperforming the S&P 500's 0.98, indicating clearly superior risk-adjusted returns. It also experiences a smaller maximum drawdown (-12.7%) compared to the benchmark (-18.9%), showing better downside protection.

Overall, incorporating news sentiment helps the strategy generate higher returns with lower risk, outperforming the market both in efficiency and drawdown control.

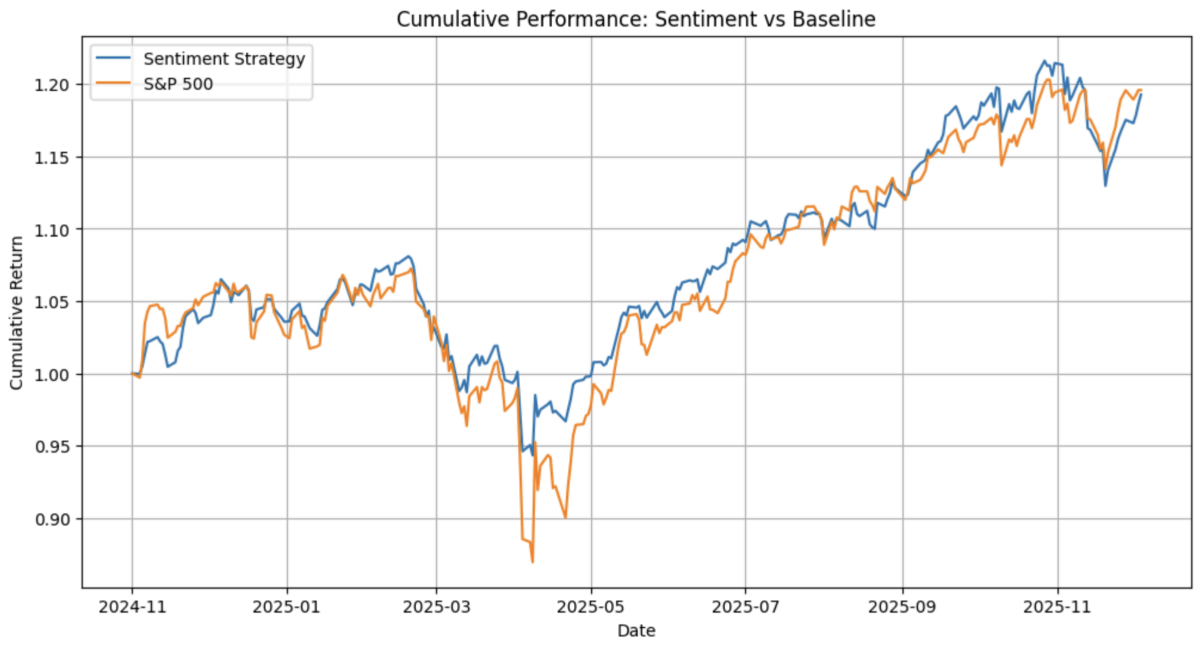

Figure1. Cumulative Returns of Sentiment-based strategy vs. S&P 500 Benchmark

IV. Lessons Learned

1. Key Takeaways For Model and Implementation Part

Be Cautious About Look Ahead Bias If Model Is Needed For Strategy Building

Initially, when building the sentiment analysis models, we randomly sampled training, validation, and test data without enforcing chronological order. The corresponding accuracy of all three sentiment analysis models are higher than the currently reported ones, with the Stacked Meta Model Vote ensemble model achieving a 0.75 accuracy on test_df.

Such results indicate that there may be some patterns or distributional shifts in later news, and if not enforcing chronological orders on training, validation, and testing data, future patterns and information leaks into the training process, leading to overly optimistic performance estimates. For any model ultimately used in the trading strategy construction, strict chronological separation is essential to avoid look-ahead bias.

Further Improving Model Performance

Our current dataset is labeled solely using ChatGPT and remains relatively small. To obtain better, more robust sentiment analysis models, we could increase the data size, incorporate manual review and correction, or use multiple LLM models to generate more consistent/accurate labels.