Market Mood & Moves: Sentiment-Driven Stock Prediction

Mentors: Meet and Sarthak

Sanjiban Paul (24B2154)

WiDS 2025-26

1 Behavioral Finance and Market Psychology

Financial markets are ever-changing and adaptive systems. The traditional approach assumes the presence of rational agents and efficient information flow, but empirical results prove that the behavior of investors follows patterns that are not entirely rational. This indicates that human behavior, overconfidence, loss aversion, as well as speculative bubbles introduce conditions of temporary inefficiency that could be capitalized on through the implementation of sentiment analysis tools; which is the objective of this project.

1.1 Quantifying Market Sentiment

Market sentiment is inferred from text data, which produces aggregate numbers as market sentiment measures. Let S_t denote the market sentiment on day t, and X_t denote traditional financial covariates such as volume or volatility. The expected return can be modeled as:

$$R_{t+1} = \alpha + \beta S_t + \gamma X_t + \varepsilon_t$$

where ε_t represents noise not captured by the model. Proper temporal alignment ensures that only information available before the market close of day t is used to predict R_{t+1}, avoiding look-ahead bias.

1.2 Behavioral Effects on Asset Prices

Investor psychology affects trading decisions:

- **Herding**: Investors follow the majority, amplifying trends.
- **Overreaction**: Prices temporarily overshoot due to news sentiment.
- **Confirmation Bias**: Traders overweight information that supports existing beliefs.
- **Loss Aversion**: Negative news has a larger impact than positive news of equal magnitude.

By quantifying these effects through sentiment analysis, trading systems can forecast short-term deviations from expected values.

2 Textual Data and NLP Techniques

2.1 Text Preprocessing

Financial text is unstructured, noisy, and domain-specific. Text preprocessing transforms text to numerical forms that can be processed by models. The usual preprocessing techniques include:

- **Tokenization**: Splitting sentences into words or subword units.
- **Stop-word removal**: Elimination of frequent words with less semantic meaning.
- **Lemmatization**: Reducing a word to its basic form, e.g., "declining" $\rightarrow$ "decline".

These steps preserve semantic meaning while reducing dimensionality.

2.2 Lexicon-Based and Model-Based Sentiment

Lexicon-based approaches give scores to words individually. For example, "profit" is positive, "loss" is negative. However, financial terms often carry domain-specific meanings:

- "Liability' is a neutral word in accounting, though negative in general English.
- "Cost" might be neutral in operations context.

Context-aware representations, for instance, transformers, are able to manage this limitation.

2.3 Aggregating Sentiment Scores

Once sentiment is extracted from individual documents, aggregation strategies include:

$$S_t = \frac{1}{N_t} \sum_{i=1}^{N_t} s_{t,i}$$

where N_t is the number of documents on day t, and $s_{t,i}$ is the sentiment score of document i. This produces a daily sentiment indicator aligned with market returns.

3 Word Representations and Embeddings

3.1 Static Embeddings

Traditional embeddings such as Word2Vec map each word w to a fixed vector $\mathbf{v}_w \in \mathbb{R}^d$. This representation is independent of the context, which can cause issues since a word can have multiple meanings: the word "bank", for example, conflates financial and geographical meanings.

3.2 Contextual Embeddings

Transformer models encode context dynamically:

$$\mathbf{v}_i = f(w_i \mid w_1, w_2, \ldots, w_n; \theta)$$

A word's representation relies on surrounding words, aiding in meaning and context; while also resolving an ambiguity. Subword tokenization (WordPiece) increases the flexibility in the treatment of out-of-vocabulary words by breaking them down, as in the case of 'Cryptoleverage'.

3.3 Positional Embeddings

Since transformers process tokens in parallel, positional embeddings E_{position} are added to encode sequential order, ensuring that the model distinguishes "Cat chases mouse" from "Mouse chases cat".

4 Transformer Architecture

4.1 Encoder Block and Self-Attention

Each Transformer layer applies self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where Q (query), K (key), and V (value) matrices are learned representations of tokens. This mechanism allows the attention of every token to every other one, capturing dependencies across long sequences.

4.2 GELU Activation

The Gaussian Error Linear Unit is used instead of ReLU:

$$\text{GELU}(x) = x\Phi(x)$$

where $\Phi(x)$ is the cumulative distribution function of a standard Gaussian. GELU introduces smooth stochasticity that improves optimization in deep networks.

4.3 Pre-training Objectives

Masked Language Modeling (MLM) Randomly mask 15% of tokens and predict them. Using the 80-10-10 strategy:

- 80% replace with [MASK]
- 10% replace with a random word
- 10% leave unchanged

Next Sentence Prediction (NSP) Predict whether sentence B follows sentence A. This helps model coherence and contextual relationships between sentences.

5 Domain Adaptation and FinBERT

5.1 Domain Shift

Financial text distributions vary greatly from generic English text distributions. Generic models can err in classifying "loss" as "share," or "share" as "loss." Domain adaptation aligns model parameters θ with financial data:

$$\theta_{\text{fin}} = \arg\min_{\theta} \mathbb{E}_{x \sim P_{\text{finance}}}[\mathcal{L}_{\text{MLM}}(x; \theta)]$$

5.2 Training Pipeline

1. General Pre-training on Wikipedia (BERT)
2. Further Pre-training on financial corpus (TRC2-Financial)
3. Supervised Fine-tuning on sentiment-labeled datasets (Financial PhraseBank)

It helps in improving accuracy and F1 value significantly compared to word embeddings or lexicon-based solutions.

5.3 Challenges

- **Catastrophic Forgetting**: Learning financial language may override general knowledge.
- **512-Token Limit**: Long documents are segmented into 512-token chunks.
- **Slanted Triangular Learning Rates**: Learning rates are increased and then decreased for efficient fine-tuning.

6 Quantitative Evaluation

6.1 Performance Metrics

- Compound Annual Growth Rate (CAGR)

$$\text{CAGR} = \left(\frac{P_T}{P_0}\right)^{1/T} - 1$$

- Maximum Drawdown (MDD)

$$\text{MDD} = \min_{t} \frac{P_t - \max_{s \le t} P_s}{\max_{s \le t} P_s}$$

- Sharpe Ratio

$$\text{Sharpe} = \frac{\mathbb{E}[R_p - R_f]}{\sqrt{\text{Var}(R_p - R_f)}}$$

6.2 Backtesting and Robust Evaluation

Backtesting using walk-forward analysis and out-of-sample data ensures that strategies are not overfit and generalize to new and unknown market conditions.

7 Sequence Modeling for Financial Time Series

Now we extend the pipeline to include temporal dependencies in financial data. Stock prices are inherently sequential, where current price movements depend not only on present information but also on historical trends, volatility regimes, and market momentum. To capture such dependencies, sequence modeling techniques are required.

Recurrent Neural Networks (RNNs) are a natural choice for modeling time-dependent data. Since classical RNNs suffer from vanishing and exploding gradient problems when trained on long sequences, we are using Long Short-Term Memory (LSTM) networks. LSTMs introduce gating mechanisms that regulate the flow of information across time steps, enabling the model to retain relevant long-term dependencies while discarding noise.

7.1 Long Short-Term Memory Networks

An LSTM cell maintains an internal memory state c_t and a hidden state h_t, updated using three gates: the input gate, forget gate, and output gate. Given an input vector x_t at time t, the LSTM updates are defined as:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

Here, $\sigma(\cdot)$ denotes the sigmoid activation function and $\odot$ represents element-wise multiplication. These mechanisms allow the network to selectively retain or forget historical

information, which is essential in modeling financial time series where market regimes may change over time.

—

7.2 Sliding Window Representation

To convert raw time series data into a supervised learning problem, a sliding window approach is adopted. Instead of predicting prices directly, the model learns a mapping from a fixed-length historical sequence to a future return value. Formally, given a sequence length L, the model learns:

$$(x_{t-L}, x_{t-L+1}, \ldots, x_t) \rightarrow y_{t+1}$$

This approach prevents information leakage by ensuring that future data is not used during training. In this project, a window length of 60 trading days was used, corresponding approximately to three months of market history.

—

8 Feature Space Design

The LSTM model operates on multivariate time series data. Each time step is represented by a feature vector combining both market-derived numerical indicators and sentiment information obtained from the natural language processing pipeline developed(Week 2.ipynb).

The numerical features include:

- Log returns
- Rolling volatility
- Relative Strength Index (RSI)
- Moving Average Convergence Divergence (MACD)
- Trading volume

In addition to these, a sentiment score derived from FinBERT is included, representing the aggregated polarity of recent financial news. This enables the model to jointly reason over quantitative price dynamics and qualitative market sentiment.

In Week 3, I trained the LSTM architecture on a seven-dimensional abstract feature space using randomly generated data. During project integration, a dummy feature was

appended to real market data to preserve architectural consistency(from week 2 to 3) while mapping meaningful financial features into the learned input space.

—

9 Model Architecture and Training

The sequence model consists of a two-layer LSTM network with a hidden dimension of 64 units. Dropout regularization is applied between layers to mitigate overfitting. The final hidden state corresponding to the last time step is passed through a fully connected layer to produce a scalar output representing the predicted next-day return.

Mean Squared Error (MSE) was used as the loss function, defined as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

The model was optimized using the Adam optimizer with a learning rate of 10^{-3}. Gradient clipping was employed to prevent exploding gradients, which are common in recurrent architectures trained on long sequences.

—

10 Integration of Sentiment and Price Dynamics

While the LSTM model outputs a numerical prediction, a raw forecast alone does not constitute a trading strategy. To bridge the gap between prediction and decision-making, a sentiment-weighted momentum rule was employed.

Let P_{LSTM} denote the predicted return from the LSTM model and S_t denote the sentiment score obtained from FinBERT for the current day. The trading decision is defined as:

$$\text{Signal} = \begin{cases} \text{BUY}, & P_{\text{LSTM}} > \epsilon \text{ and } S_t > \tau \\ \text{SELL}, & P_{\text{LSTM}} < -\epsilon \text{ and } S_t < -\tau \\ \text{HOLD}, & \text{otherwise} \end{cases}$$

where ϵ and τ are empirically chosen thresholds controlling sensitivity to predicted returns and sentiment confidence respectively.

This rule ensures that trades are executed only when both technical momentum and fundamental sentiment align, thereby reducing exposure to spurious predictions.

—

11 End-to-End Project Integration

The complete system integrates all components developed across the project timeline. Historical price data is sourced from public financial APIs, while real-time financial news is obtained using NewsAPI. Sentiment analysis is performed using a FinBERT transformer model, and the resulting sentiment scores are aligned temporally with price data.

The integrated pipeline operates as follows:

1. Fetch historical OHLCV data for selected stocks.
2. Compute technical indicators and rolling statistics.
3. Retrieve recent financial news and compute sentiment scores using FinBERT.
4. Construct fixed-length sequences using a sliding window.
5. Generate return predictions using the trained LSTM model.
6. Apply the sentiment-weighted trading rule to produce BUY, SELL, or HOLD signals.

The final output is a structured report summarizing predicted returns, sentiment scores, and recommended actions for each stock. This modular design separates prediction from decision logic, aligning with best practices in quantitative finance systems.

—

12 Discussion and Limitations

The Week 3 implementation demonstrates the feasibility of combining transformer-based sentiment analysis with recurrent neural networks for financial time series modeling. However, several limitations remain. The LSTM model was initially trained on synthetic data to validate architectural correctness, and future work will involve retraining the model on real historical features for improved predictive performance.

Additionally, transaction costs, slippage, and market impact were not considered in the current trading logic. Incorporating these factors, along with backtesting and risk-adjusted performance metrics such as the Sharpe ratio, would be necessary for deployment in real-world trading environments.

—

13 Conclusion

This project successfully extends the sentiment analysis framework developed earlier into a complete, end-to-end decision-making system. By integrating natural language processing, time series modeling, and rule-based trading logic, the project demonstrates a multidisciplinary approach to financial prediction. The modular pipeline developed here provides a strong foundation for future extensions, including attention-based architectures, portfolio-level optimization, and comprehensive backtesting.

14 References

- GeeksforGeeks website
- Wall Street Quants YT Channel
- https://www.bavest.co/en/post/market-psychology-and-sentiment