Examining the Effect of News Context on Algorithmic Trading

Surupendu Gangopadhyay and **Prasenjit Majumder**
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, India, 382007
{surupendu.g, p_majumder}@daiict.ac.in

Abstract

The application of reinforcement learning in algorithmic trading for spot trading, wherein the state is represented using price data, is a well-explored problem. However, these works perform spot trading in an index, which is not the market norm. Recent works have explored the use of reinforcement learning for performing high-frequency trading in the futures market. These works also combine news data and price data to represent the state. However, the news data is represented using news sentiment, which is not the optimal solution for representing the contextual information of news data. This paper proposes an RL framework that factors in the contextual information of the news data by using text embedding models and combines this with price data to perform futures trading. The paper investigates the impact of using different text embedding models on the trading strategy of the RL agent. Further, the paper also investigates whether using news context representation improves the trading decisions of the proposed model. The models are evaluated on NIFTY 50 index. The evaluation metrics show that using news contextual representation to represent the news data improves the trading performance of the RL agent.

1 Introduction

Reinforcement learning (RL) in algorithmic trading involves using trading agents that detect and leverage hidden signals in numerical and non-numerical data sources to open or close a position in the market. The goal of the agent is to learn an optimal trading strategy that maximizes the profit of the agent. In reinforcement learning, the agent interacts with the environment and modifies its policy based on the reward it receives from the environment. Additionally, the agent uses an exploration-exploitation strategy to learn the actions that eventually lead to the optimal policy. This property enables the agent to operate in unseen conditions as well. Thus, in the case of using reinforcement learning for algorithmic trading, the agent can learn a dynamic trading strategy to operate in the highly temporal stock market environment.

The current literature on reinforcement learning for algorithmic trading widely uses price data, a numerical data source for designing the RL agent. The price data consists of historical OHLCV values and technical indicator values (Jeong and Kim, 2019; Wu et al., 2020; Théate and Ernst, 2021; Taghian et al., 2022; Yang et al., 2023). In these works, the trading agent primarily uses DQN (Mnih et al., 2015), which is an off-policy based RL algorithm. The trading agent performs spot trading and operates only once a day before the market closes. Thus the agent operates in a less noisy environment as the market trend already reflects the activity of the other stakeholders in the market. Also, these works directly perform spot trading instead of futures trading in an index, which is not the market norm.

Recent works have aimed to combine news data, which is a non-numeric data source, and price data to represent the state of the environment to perform trading in the share market (Gangopadhyay and Majumder, 2023a). The authors in this work represent the news data using the news sentiment. The agent operates in a noisy environment as the other stakeholders are also active in the market. The authors compare the performance of RL agents that use DQN and PPO (Schulman et al., 2017) which is an on-policy based RL algorithm. The authors show that in such conditions using PPO as the RL agent is well-suited as it uses the samples of the current trajectory to update the policy.

However, using news sentiment to represent the news data can give false positive and false negative signals from the news articles, which can impact the performance of the trading agent. Thus, using news sentiment to represent the contextual information of the news data is not an optimal solution,

Proceedings of the Joint Workshop of the 8th Financial Technology and Natural Language Processing (FinNLP) and the 1st Agent AI for Scenario Planning (AgentScen), Jeju, South Korea, August 3, 2024

and we need to explore the use of text embedding models to embed the context of the news data and use these news embeddings along with price data for representing the state. Therefore, based on this we investigate the following research questions:

- RQ1: How an RL agent can factor in the contextual representation of news data and combine it with price data to make trading decisions?
- RQ2: Can using contextual representation of news data along with price data improve the trading decisions of an RL agent?

In this paper, we propose an RL framework that leverages the contextual information of the news data and price data to perform futures trading at a high frequency. We use text embedding model to embed the news articles in the news data and use these embeddings along with price data to represent the state. Based on the findings of Gangopadhyay and Majumder (2023a) we use PPO as the RL agent. We use a feature extraction module to extract features from the state. In this work, we also compare the effect of using different text embedding models on the trading performance of the RL agent. We evaluate the effect of using different text embedding models by trading in the NIFTY 50 index. Our experiments show that factoring in the news context leads to an improvement in the trading performance of the model. The code repository of this work is available here.

2 Related Work

The current literature on using reinforcement learning for algorithmic trading primarily focuses on spot trading in the stock market wherein the agent takes an action only once in a day before the market closes. Though some works have explored using reinforcement learning for algorithmic trading in the futures market. In the off-policy based approach the authors use DQN as the RL agent (Jeong and Kim, 2019; Wu et al., 2020; Théate and Ernst, 2021; Taghian et al., 2022; Yang et al., 2023). In the on-policy based approach the authors use policy gradient (Lei et al., 2020), PPO (Hirchoua et al., 2021; Gangopadhyay and Majumder, 2023a), deterministic policy gradient (Wu et al., 2020) as the RL agent. Recent works have also explored RL agent that uses an ensemble of on-policy and off-policy based RL algorithms (Yang et al., 2020; AbdelKawy et al., 2021).

In some works the authors use a discrete action space wherein the RL agent can perform three actions $[+1, -1, 0]$ (Jeong and Kim, 2019; Hirchoua et al., 2021; AbdelKawy et al., 2021; Lei et al., 2020) indicating buy, sell and hold or two actions $[-1, +1]$ (Taghian et al., 2022; Yang et al., 2023; Théate and Ernst, 2021) indicating buy and sell. The use of discrete action space leads to the curse of dimensionality, which reduces the scalability of the RL agent to trade with a variable number of shares. Thus, in the case of discrete action space, the authors generally trade in a fixed number of shares. The papers that use on-policy based RL agents use a continuous action space (AbdelKawy et al., 2021; Gangopadhyay and Majumder, 2023a) wherein the action lies between $[-1, +1]$, which overcomes the limitations of using a discrete action space.

The works that use only price data to represent the state use the OHLCV values, technical indicators, stock trends, account balance as the price data. Generally, the authors use the raw values of the price data of the previous day to represent the state. However, some works consider the price data as a time series and thus use a window of price data to represent the state. These works use encoder models such as GRU (Wu et al., 2020; Taghian et al., 2022), attention-based GRU (Lei et al., 2020), transformers (Yang et al., 2023), CNN1D, CNN2D (Taghian et al., 2022) to extract the features from the sequence. Taghian et al. (2022) compare the performance of RL agents when using raw OHLC values and a window of OHLC values to represent the state and show that using a feature extraction module can improve the performance of an RL agent. Gangopadhyay and Majumder (2023a) perform HFT in the futures market using a combination of news data and price data to represent the state. The authors represent the news data using news sentiment and a feature extraction module that uses CNN to extract features over a window of prices and news sentiments.

In the current literature the reward function generally use the relative difference between the close prices or the difference in close prices as the reward function. The authors evaluate the trading models using evaluation metrics such as total profit, Sharpe ratio, Sortino ratio, maximum drawdown, and returns.

3 Proposed Approach

We propose an RL framework that trades in futures contracts in a minute-wise time series setting. At each time step t the agent uses the state s_t and reward r_t to determine the action a_t that will be executed at the next time step $t+1$. The RL agent leverages the hidden signals in the news data and price data to determine the action. In this approach we consider all contracts as near month contracts, so the contracts will expire at the last Thursday of every month when the market closes. The agent can take a long or short position in the market and carry forward a position to the next day. When a contract expires all the open positions of the agent are automatically closed. We train the agent based on episodes, wherein each episode is the duration between the start and expiry of a contract. We use the environment used in Gangopadhyay and Majumder (2023a) to simulate the execution of buying and selling of futures contracts. We describe in detail the components of the RL framework in the subsections 3.1, 3.2, 3.3, and 3.4.

3.1 State

We use news data and price data to represent the state (s_t). We represent the price data at time step t by using the technical indicator values from time steps $t-w$ to t where w is the window size. The technical indicator values [1] consist of ADX, MACD, MOM, ATR, RSI, Slow %K, Williams %R, BBAND, EMA. At each time step i ($i \in [t-w, t]$), we construct a price vector denoted as $price_i$ which consists of these technical indicator values. Thus, the price data is a sequence of price vectors denoted as $[price_{t-w}, ..., price_t]$.

We represent the news data at time step t by using k latest news article titles published between time steps $t-w'$ to t where w' is the window size. Thus, the news data is represented as a sequence of news article titles $[news_1, news_2, ..., news_k]$. We use text embedding model to embed the context of a news article title and represent a news article title $news_j$ ($j \in [1, k]$) using a news embedding denoted as n_j. Thus, the news data is a sequence of news embeddings denoted as $[n_1, ..., n_k]$. The state s_t is represented as combination of sequence of price vectors and news vectors.

As we are examining the effect of using different text embedding models on the performance of the trading agent, so in this work we compare the performance of the trading agent when using transformer encoder-based and transformer decoder-based text embedding models. In transformer encoder-based embedding models we use BERT (340M parameters) (Devlin et al., 2018) and FinBERT (340M parameters) (Araci, 2019). LLMs can provide a richer representation of the text due to a considerable increase in the number of the trainable parameters of the model and amount of data on which it is pretrained. All the major LLMs use the transformer decoder for text representation. So in transformer decoder-based embedding models we use Llama 2 (7B parameters) (Touvron et al., 2023) and Mistral (7B parameters) (Jiang et al., 2023).

As we are using a GPU resource poor setup for this work we do not finetune these text embedding models w.r.t. to our trading task rather we directly use the use these models in inference mode to get the embeddings. In BERT and FinBERT we take a sum of the token embeddings at the last layer to represent a news title. In Llama 2 and Mistral we use the embedding of last token in the sequence to represent a news title. Further, in this work we use 4-bit quantized AWQ (Lin et al., 2023) versions of Llama 2 and Mistral to enable inference of these models in the GPU resource poor setting.

3.2 Action

The action is the number of lots that agent intends to buy or sell. We use a continuous control setting in our proposed approach, so we define action space as $\mathcal{A} \in [-1, +1]$. The values of the action a_t lies within this action space. We define the maximum number of lots (max_lots) that the agent can buy or sell and use Equation 1 to get the actual number of lots that the agent wants to buy or sell. The use of a discrete action space will lead to the curse of dimensionality if we increase the number of lots that the agent can buy or sell, where as the continuous action space allows the agent to scale easily to trade in a high number of lots.

$$a_t = \lfloor max\_lots \times a_t \rfloor \tag{1}$$

3.3 Agent

The agent uses PPO which is an on-policy based RL algorithm. As the state consists of a sequence of news vectors and price vectors, so the agent uses a feature extraction module to extract features from the state s_t and form a feature vector denoted as

[1] https://www.incrediblecharts.com/indicators/technical-indicators-az.php

f_t. PPO uses this feature vector f_t to determine the action a_t. The feature extraction module is shown in Figure 1.

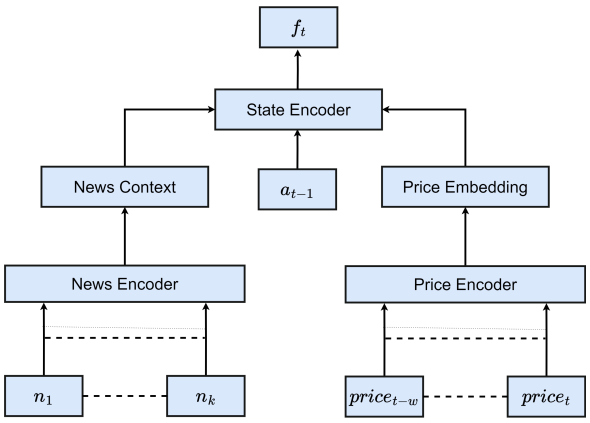

Figure 1: Architecture of feature extraction module

The feature extraction module consists of the following modules: news encoder, price encoder and state encoder. The news encoder module uses a CNN layer to capture the local contextual relationship between the sequence of news vectors to get the context vectors $[c'_1, ..., c'_k]$ as output. It then uses Equation 2 to form a news sequence vector denoted as ns_v.

$$ns_v = \sum_{k}^{j=1} c'_j \tag{2}$$

It then passes ns_v through two fully connected neural layers to get a single value as output and applies a sigmoid function over the single value to get the news context value denoted as nc_v ($nc_v \in [0, 1]$) which represents the context encoded in the sequence of news vectors.

The price encoder module uses a CNN layer to capture the local contextual relationship between the sequence of price vectors to get the context vectors $[c_{t-w}, ..., c_t]$ as output. It then uses Equation 3 to form a price sequence vector denoted as p_v. It then passes p_v through a fully connected neural layer to get the price embedding denoted as p_e which encodes the contextual relationship of the price vectors.

$$p_v = \sum_{i=t-w}^{t} c_i \tag{3}$$

The state encoder module concatenates nc_v, p_v and previous action take by the agent (a_{t-1}) and passes it through a fully connected neural layer to form the feature vector f_t. We use CNN in the news encoder and price encoder, as it can effectively capture the context of a sequence (Gangopadhyay and Majumder, 2023b).

PPO consists of policy network and value network wherein the policy network determines the action and the value network gives the expected returns associated with the action. Both the networks consist of two fully connected neural layers which use the feature vector f_t to determine the action and value function. The policy network and value network share the parameters of the feature extraction module.

3.4 Reward Function

The reward function considers the short-term reward of an action w.r.t. change in close prices from t to $t+1$ and long-term reward of an action w.r.t. change in the balance of the agent from t to $t+1$ to calculate the reward denoted as r_t (Gangopadhyay and Majumder, 2023a). We use Equation 4 to calculate r_t where, c_t indicates the close price at time step t, $balance_t$ indicates the balance of the agent at time step t and a_t is the value that we get from Equation 1. The equation uses a λ value to balance the short-term and long-term rewards.

$$\begin{aligned} r_t = \lambda \times (a_t \times (c_{t+1} - c_t)) \\ + (1 - \lambda) \times (balance_{t+1} - balance_t) \end{aligned} \tag{4}$$

4 Dataset

The dataset consists of news data and price data. The news data comprises of archive news articles from the Economic Times [2] from 2010-2021 . We use a proprietary classifier to select only financial news articles from the news articles. The price data consists of minute-wise OHLC prices of NIFTY 50 index [3] from 2010-2021. We calculate the technical indicators values mentioned in the proposed approach from these prices and perform a z-normalization over the technical indicator values.

5 Evaluation Metrics

1. Total Profit: It is the profit earned at the end of the trading session. It is the difference between the balance earned at the end of the trading session and the balance at the start of the trading session.

[2] https://economictimes.indiatimes.com/archive.cms?from=mdr

[3] https://www.kaggle.com/datasets/nishanthsalian/indian-stock-index-1minute-data- 2008-2020

2. Return (%): It is the percentage of the relative difference between the balances at the end of the trading session and start of the trading session.

3. Maximum Drawdown (MDD): MDD indicates the maximum loss a trading agent can incur in a trading session. It calculates this by measuring the relative difference the highest peak and lowest trough before the next peak is achieved. The duration between the two peaks indicates the time taken by the agent to recover from the loss.

4. Volatility: Volatility indicates the risk associated with an investment during a trading session. It is measured using the variance (σ) of the daily returns times the number of trading days (D) in a trading session. Volatility is calculated using Equation 5.

$$\text{`Volatility} = \sigma\sqrt{D} \tag{5}$$

5. Sharpe Ratio: Sharpe Ratio is the ratio of expected returns $(E(R))$ in a trading session and the volatility in the trading session. Sharpe Ratio is calculated using Equation 6, wherein we assume the risk free rate to be zero.

$$\text{Sharpe Ratio} = \frac{E(R)}{\sigma} \tag{6}$$

6. Sortino Ratio: Sortino Ratio is the ratio of expected returns in a trading session and the standard deviation of the negative returns (σ_d) in a trading session. Sortino Ratio is calculated using Equation 7, wherein we assume the risk free rate to be zero.

$$\text{Sortino Ratio} = \frac{E(R)}{\sigma_d} \tag{7}$$

6 Experimental Setup

In this work we use Nvidia-RTX 20280Ti GPU consisting of 11GB VRAM. We use the data from 2010-2016 for training the trading models and we evaluate the models over the years 2017-2021. As per the market rules we set the size of a single lot for the years 2010-2017 to 25 and from 2018-2021 to 75. The maximum number of lots (max_lots) that agent can trade is 3. The starting balance for each test year is shown in Table 1, the balance is the product of the maximum number of lots that agent can trade and the share price at the start of the trading session. We set the maximum sequence length of a news article title to 40 tokens. At each time step t we consider the news articles published in last 60 mins and thus set the value of w' to 60 and select the latest 10 news articles published within this window, thus we set k to 10. The graphs indicating the occurrence of sequence of news articles in the 1 hour window for training data and test data are shown in Figures 2 and 3. From the graphs we can observe that we are dealing with sparse data while dealing with news data. In this work, we consider the price data of last 5 minutes and thus set the value of w to 5. The λ value in the reward function is set to 0.85 (Gangopadhyay and Majumder, 2023a). The PPO algorithm uses Adam optimizer to perform optimization of the neural network layers. The parameters of policy network and value network of PPO and the parameters of the feature extraction module are given in Appendix A.1. The hyperparameters for training the different models is given Appendix A.2.

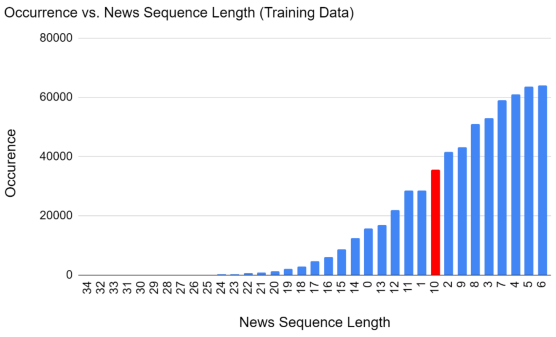

Figure 2: The frequency of occurrence of the length of the sequence of news articles present within 1 hr window (Training data)

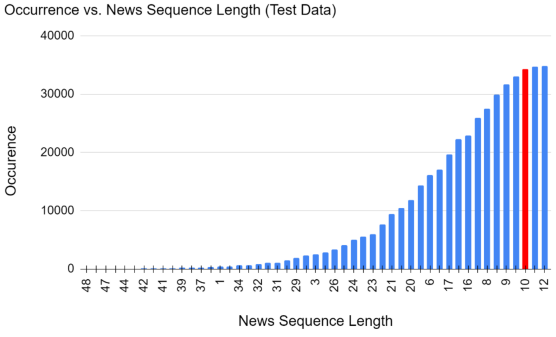

Figure 3: The frequency of occurrence of the length of the sequence of news articles present within 1 hr window (Test data)

Year	2017	2018	2019	2020	2021
Initial Balance	615757.5	2369632.5	2448382.5	2745483.75	3149122.5

Table 1: Initial balance at start of each test year

7 Results

In this section, we compare the trading performance of the proposed trading model when we use different text embedding models to embed the news data. As mentioned in the section 3.3, we compare the trading model w.r.t. BERT (768 dimension news embedding), FinBERT (768 dimension news embedding), Llama 2 (4096 dimension news embedding), and Mistral (4096 dimension news embedding). The year-wise total profit and average total profit is shown in Table 2. The year-wise return (%) and average return (%) is shown in Table 3. The year-wise MDD (%) and MDD duration (days) and average MDD (%) and average MDD duration (days) are shown in Table 4 and Table 5 respectively. The average volatility, Sharpe Ratio and Sortino Ratio are shown in Table 6, Table 7 and Table 8 respectively.

	Total Profit			
Years	**BERT**	**FinBERT**	**Llama 2**	**Mistral**
2017	229038.31	246737.37	152081.5	433339.06
2018	1083313.68	1154277.93	1235783.81	1314911.06
2019	1393744.87	1750031.24	1653581.06	936973.68
2020	3010310.81	3735349.31	3914199.37	3603385.68
2021	2008915.68	1240028.06	3296598.74	2266329
Avg. Profit	1545064.67	1625284.78	**2050448.89**	1710987.7

Table 2: Total Profit of the proposed trading model when using different text embedding models

Compared to the other text embedding models, we observe that using Llama 2 for news embedding gives the highest total profit and return (%), followed by Mistral, FinBERT, and BERT. Using FinBERT for news embedding gives the lowest MDD (%) and MDD duration of 26.277 % and 17.8 days, respectively, whereas using Llama 2 gives the highest MDD (%) and MDD duration of 27.813 % and 38.8 days respectively. We observe that the trading model faces similar loss percentages when using FinBERT and Llama 2, but the duration for which the trading model is at a loss is lower in FinBERT than in Llama 2. In terms of volatility, which indicates the risk of an investment,

	Return (%)			
Years	**BERT**	**FinBERT**	**Llama 2**	**Mistral**
2017	37.19	40.07	24.69	70.37
2018	45.71	48.71	52.15	55.49
2019	56.92	71.47	67.53	38.26
2020	109.64	136.05	142.56	131.24
2021	63.79	39.37	104.68	71.96
Avg. Return (%)	62.65	67.13	**78.32**	73.46

Table 3: Return (%) of the proposed trading model when using different text embedding models

Llama 2 has a volatility of 2.18, which is lower than BERT by 0.191, lower than Mistral by 0.181, and higher than FinBERT by 0.267. We also observe that using BERT gives the highest volatility and lowest return (%), indicating that it takes risky actions which yield lower returns. The Sharpe ratio and Sortino ratio of Llama 2 and Mistral are higher than BERT and FinBERT and thus indicate that using LLMs increase the volatility of the trading model but they also yield higher returns. The return (%) and MDD show that using FinBERT for embedding the news data improves the performance of the trading model than using BERT. Thus we infer that finetuning a pretrained language can improve the trading performance as it can better represent the context of a financial news article.

	MDD (%)			
Years	**BERT**	**FinBERT**	**Llama 2**	**Mistral**
2017	29.76	26.41	33.62	30.245
2018	27.01	25.77	27.6	26.09
2019	28.14	26.48	26.97	27.17
2020	26.19	24.89	25.88	25.81
2021	27.61	27.82	24.96	27
Avg. MDD (%)	27.74	**26.27**	27.81	27.26

Table 4: MDD (%) of the proposed trading model when using different text embedding models

Since, Llama 2 and Mistral are LLMs with higher model parameters and are pretrained on more text data than BERT and FinBERT. These factors lead to a more robust representation of the news data, thus improving the trading performance of the RL agent. Thus, using LLMs for text em-

	MDD Duration (Days)			
Years	**BERT**	**FinBERT**	**Llama 2**	**Mistral**
2017	30	14	147	50
2018	21	2	9	25
2019	34	9	24	29
2020	5	48	9	5
2021	16	16	5	39
Avg. MDD (Days)	21.2	**17.8**	38.8	29.6

Table 5: MDD duration (days) of the proposed trading model when using different text embedding models

bedding leads to higher returns accompanied with high investment risk, which corresponds with the efficient market hypothesis which states that to get higher returns, we need to take higher risk in investments (Fama, 1970). In the current literature Mistral (7B) has outperformed Llama 2 (13B) in various tasks (Jiang et al., 2023) whereas in this work, we find that the total profit and return (%) of Llama 2 are higher than that of Mistral by 3,39,461.199 and 4.858 %, respectively. However, the MDD (%) and MDD duration of Mistral is lower than Llama 2 by 0.547 % and 9.2 days. The performance of Llama 2 improves when using AWQ 4-bit quantization (Lin et al., 2023), but in this case, using AWQ 4-bit quantization may have affected the performance of Mistral.

However, the results show that the proposed approach can combine the contextual representation of the news data with price data to detect hidden signals in the data and exploit it to perform HFT in the futures market, thus answering our RQ1.

	Volatility			
	BERT	**FinBERT**	**Llama 2**	**Mistral**
Avg. Volatility	2.37	**1.91**	2.18	2.36

Table 6: Volatility of the proposed trading model when using different text embedding models

7.1 Comparison with existing approaches

Based on the return (%), we select the model that uses Llama 2 for news embedding as our best-performing model. We term this model as PPO_Llama_2. We compare this model with PPO_P, which uses only price data, i.e., a sequence

	Sharpe Ratio			
	BERT	**FinBERT**	**Llama 2**	**Mistral**
Avg. Sharpe Ratio	0.08	0.07	0.08	0.08

Table 7: Sharpe ratio of the proposed trading model when using different text embedding models

	Sortino Ratio			
	BERT	**FinBERT**	**Llama 2**	**Mistral**
Avg. Sortino Ratio	0.15	0.11	0.14	0.15

Table 8: Sortino ratio of the proposed trading model when using different text embedding models

of technical indicator values to represent the state, and PPO_FEM_PT, which uses price data and sentiment of news titles in the news data to represent the state (Gangopadhyay and Majumder, 2023a). We compare these models based on average return (%), average MDD (%), and average MDD duration. The comparison of the results of these models are shown in Table 9.

Trading Model	**Return (%)**	**MDD (%)**	**MDD Duration (Days)**
PPO_P	25.75	-26.81	47.6
PPO_FEM_PT	52.81 (+27.06)	-29.68 (-2.87)	41.6 (-6)
PPO_Llama_2	78.32 (+25.51)	-27.81 (+1.87)	38.8 (-2.8)

Table 9: Comparison of trading models based on Avg. Return (%), Avg. MDD (%) and Avg. MDD duration (days)

We observe that the return (%) of PPO_Llama_2 is higher than PPO_FEM_PT by 25.51 % and higher than PPO_P by 52.576 %. The duration for which PPO_Llama_2 faces a loss is less than PPO_FEM_PT by 2.8 days and less than PPO_P by 8.8 days. We observe that using the contextual representation of the news articles and a CNN layer to discover the relationship between the context of the sequence of news articles leads to a better representation of news data than using the aggregate of sentiments of news articles to represent the news data. We infer that news sentiment may not always convey the actual intent of a news article title and lead to generating false signals from the news data. Factoring in the news context reduces the false sig-

nals and greatly improves the performance of the RL agent and hence answers our RQ2.

8 Conclusion and Future Work

The results of the proposed trading model show that using news data context along with price data leads to an improvement over the model that uses news data sentiment along with price data or uses only price data. The results of FinBERT show that domain specific language models lead to an improvement in returns while also reducing the MDD. Increasing the text embedding model parameters improves the trading performance of the trading model, as we observed with the use of LLMs. Thus, in the future, we need to finetune the LLMs on financial texts and use this finetuned LLM to improve the trading performance of the RL agent. Future studies should also focus on using the article body instead of only the news title.

In this work, we deal with sparsity in availability of news as at time step t we used news data published in the last 60 mins preceding time step t. Increasing the number of hours led to non-convergence in training of the trading model. This sparsity affects the performance of the RL agent. The use of news data as the source of non-numeric data may also lead to some lag between when the information was available and when it was published, which may have affected the trading performance. In future work, we need to explore the use of multimodal news sources to bridge over the sparsity and time lag of news articles. Also, factoring in longer context length using the news article body instead of the title needs to be examined for better contextual representation. Future research should also focus on the adversarial training of the trading agents to guard against fake news, which can adversely affect the performance of the trading agent.

References

Rasha AbdelKawy, Walid M Abdelmoez, and Amin Shoukry. 2021. A synchronous deep reinforcement learning model for automated multi-stock trading. *Progress in Artificial Intelligence*, 10(1):83–97.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Eugene F Fama. 1970. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417.

Surupendu Gangopadhyay and Prasenjit Majumder. 2023a. How does news data impacts trading decisions? *Available at SSRN 4551629*.

Surupendu Gangopadhyay and Prasenjit Majumder. 2023b. Text representation for direction prediction of share market. *Expert Systems with Applications*, 211:118472.

Badr Hirchoua, Brahim Ouhbi, and Bouchra Frikh. 2021. Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. *Expert Systems with Applications*, 170:114553.

Gyeeun Jeong and Ha Young Kim. 2019. Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Kai Lei, Bing Zhang, Yu Li, Min Yang, and Ying Shen. 2020. Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading. *Expert Systems with Applications*, 140:112872.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Mehran Taghian, Ahmad Asadi, and Reza Safabakhsh. 2022. Learning financial asset-specific trading rules via deep reinforcement learning. *Expert Systems with Applications*, 195:116523.

Thibaut Théate and Damien Ernst. 2021. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti

Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xing Wu, Haolei Chen, Jianjia Wang, Luigi Troiano, Vincenzo Loia, and Hamido Fujita. 2020. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538:142–158.

Bing Yang, Ting Liang, Jian Xiong, and Chong Zhong. 2023. Deep reinforcement learning based on transformer and u-net framework for stock trading. *Knowledge-Based Systems*, 262:110211.

Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. 2020. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–8.

A Appendix

A.1 Model Configuration

The configuration of the news encoder in the feature extraction module for different text embedding models is shown in Table 10. The configuration of the price encoder and state encoder in the feature extraction module for different text embedding models is shown in Table 11. The CNN layer uses a kernel size of 3 for all models. The parameters of policy network and value network of PPO is shown in Table 12.

	News Encoder		
Text Embedding Model	**CNN**	**Layer 1**	**Layer 2**
BERT	768×200	200×100	100×1
FinBERT			
Llama 2	4096×1000	1000×500	100×1
Mistral			

Table 10: Parameters of news encoder in the feature extraction module

	Price Encoder		**State Encoder**
Text Embedding Model	**CNN**	**Layer 1**	**Layer 1**
BERT	14×14	14×14	16×16
FinBERT			16×64
Llama 2			16×128
Mistral			16×16

Table 11: Parameters of price encoder and state encoder in the feature extraction module

	PPO	
Text Embedding Model	**Policy Network**	**Value Network**
BERT	16×16	
FinBERT	64×16	
Llama 2	64×64	
Mistral	64×64	

Table 12: Parameters of policy network and value network of PPO

A.2 Hyperparameters

The hyperparameters for training the models are shown in Table 13.

Text Embedding Model	**BERT**	**FinBERT**	**Llama 2**	**Mistral**
Learning Rate	0.0002	0.0002	0.00019	0.00019
Batch Size	128	128	128	128
Entropy Co-efficient	0.02	0.02	0.02	0.02
Epochs	6	6	7	6
Steps	2000	1500	1500	1500

Table 13: Hyperparameters for training the models