**CONTENTS:**

|  |  |
| --- | --- |
| Topics | Page number |
| Abstract | 1 |
| Related Work | 1 |
| 1. Introduction | 1 - 2 |
| 1. Reinforcement Learning | 2 - 3 |
| 1. Q-Learning | 3 |
| 1. Proposed system of RL with sentiment scores | 4-5 |
| *4.1 Input* | 4-5 |
| *4.1.1 Stock Data* | 5 |
| * + 1. *Sentiment Scores* | 5 |
| 1. RL Network | 5 |
| 1. Deep neural network | 5 - 6 |
| 1. Training of RL Agent | 6 |
| 1. Experience replay | 6 - 8 |
| 1. Results and Discussion | 8 - 11 |
| *9.1 Dataset* | 8 |
| *9.2 Reinforcement Learning (RL) on Dataset* | 9 - 11 |
| *9.2.1 Training with RL Without Sentiment Input* | 9 |
| *9.2.2 Testing with RL Without Sentiment*  *Input* | 9 - 10 |
| * + 1. *Training with RL with Sentiment Input* | 10 |
| *9.2.4 Testing with RL with Sentiment Input* | 10 - 11 |
| 1. Comparison between with and without sentiment scores: | 1. - 12 |
| *10.1 Portfolio results from testing data comparisons* | 12 |
| 1. Conclusions and Future Works | 12 |
| 1. Appendix | 12 |

**Stock Trading Agent Using Reinforcement Learning**

**Abstract:**

 The stock market is a place, where shares of different companies are traded. It is a collection of buyers’ and sellers’ stocks. In this digital era, analysis and prediction in the stock market have gained an essential role in shaping today's economy. Stock market analysis can be either fundamental or technical. Technical analysis can be performed either with technical indicators or through machine learning techniques. In this project, I report a system that uses a Reinforcement Learning (RL) network and market sentiment score to make decisions about stock market trading. The system uses sentiment analysis on daily market news to spot trends in stock prices. The compound scores in the dataset have been generated using sentiment analysis on daily market news. This score is fed into the RL module as one of its inputs. The RL section gives decisions in the form of three actions: buy, sell, or hold. The objective is to maximize long-term future profits. I have used stock data of Apple from 2006 to 2016 to interpret how sentiments affect trading. The stock price of any company rises, when significant positive news become available in the public domain. The results obtained reveal the influence of market sentiments on the forecasting of stock prices.

**Related Work:**

One of the classic stock price forecasting methods is the Autoregressive Integrated Moving Average (ARIMA). The combination of past errors and past values is considered in ARIMA to form the future value. This model depends on the close price, high price, low price, and open price to predict future values. Even though the ARIMA model is efficient in forecasting time series data, it cannot be used if the data contains a seasonal component. The time-series data is said to have a seasonal component if it contains repetitive cycles. For the modeling of such data, an advanced ARIMA model is known as SARIMA. The SARIMA model is effective in financial forecasting, specifically for the short and medium ranges. The traditional models, such as ARIMA and SARIMA can be used effectively only on stationary time series data. In a live trading scenario, implementing these models is a great challenge, because the new coming data may not always be stationary time series data.

1. **Introduction**

The stock market is a platform where sellers and buyers of stocks interact. It allows people to access stock exchanges through their computers and conduct transactions, mainly buying and selling stocks. Stock exchanges facilitate trading by providing a real-time interface between buyers and sellers, allowing a systematic matching process between willing buyers and sellers. Investing in stocks that are present in the stock market can lead to financial gains. Analyzing stocks is crucial in deciding whether to buy, sell or hold a stock. Efficient and accurate stock price prediction can lead to significant profits. Studying the behavior of financial markets is a great challenge. However, well-researched and accurate perspectives, which include both directional views and information such as the price of stocks, expected risk, and expected reward, can aid in stock market analysis.

Fundamental analysis and technical analysis help in developing a good view of the market. Fundamental analysis involves researching a few companies and making future predictions based on their performance. On the other hand, the technical analysis considers current market trends and scans the market for opportunities. Technical indicators are used for this analysis.

One must understand the difference between the two types of analysis to succeed in the stock market. Fundamental analysis is preferred in long-term investments, and technical analysis is chosen for short-term investments. An investor can earn a small yet consistent profit frequently in a short term by choosing trading decisions with technical analysis. The application domains of both methods of analysis also differ. Technical analysis can be applied to all asset classes like equities, commodities, income, etc. In contrast, fundamental analysis is specific to each asset class.

To find stock chart patterns for technical analysis, predict market prices, and make decisions, machine learning methods are widely used. Deep learning can be used to find hidden patterns in stock data. The Deep Learning Neural Network (DLNN) is a suitable model for multivariate time series analysis due to its built-in properties. This model is robust to noise in input data, can support learning and predictions even with missing values, and can learn both linear and non-linear relationships in data.

However, even if we can predict prices by certain patterns and make decisions using the above-mentioned methods, there is no way to learn from the decisions made and to make better decisions in the future by learning from past actions. This is where RL comes into the picture of stock market trading systems. RL is a machine learning technique in which the system learns from the environment and tries to maximize rewards.

The main contributions of this work are:

1. The inclusion of market sentiments in an RL-based system used for stock market trading.
2. The performance comparison of an RL network with sentiments and an RL network without sentiments.
3. The investigation of the influence of market news on deciding stock prices.
4. **Reinforcement Learning (RL)**

Reinforcement Learning (RL) is a branch of machine learning that aims to learn how to take suitable actions to maximize reward in a particular situation. RL networks consist of four main components: environment, agent, reward, and policy. The policy is the solution for the RL problem, which maps each state of the environment to a specific action.

Figure 1 provides an overview of the RL technique. In RL, an agent interacts with its environment in discrete time periods, receiving an observation (St) at each time step, which typically includes the reward (Rt). The agent then selects an action from a set of existing actions, and this action is transmitted to the environment, resulting in a new state (S(t+1)) and a new reward (R(t+1)). The goal of an RL agent is to accumulate as much incentive as possible.

The mathematical formulation of RL problems is called the Markov Decision Process (MDP), where the current state depends only on the previous state. The solution for an MDP is defined using policies. A policy is a set of actions that the agent takes to reach a goal. It can be denoted as π, and represented as π(s)→a, where a ∈ A and s ∈ S. A policy can also be stochastic, denoted as π(s,a)→P(s). P(a|s) represents the probability of taking action a in state s, where a ∈ A and s ∈ S. An optimal policy that maximizes the reward is denoted as π\*, which has the optimal actions for all states to minimize future rewards. To find the optimal actions for each state, an algorithm called Q-learning is used.

Diagram

Description automatically generated

Fig 1: Reinforcement Learning

1. **Q-Learning**

In Q-learning, we define a function, Q(s, a), representing the discounted future reward when we perform action a in state s, and continue optimally from that point on:

Q(st, at) = maxπ R(t+1)………. (1)

Equation (1) gives the Q-value function, which represents the quality of a certain action in a given state, i.e., the best possible reward at the end of the game after performing action a on state s at time t, by following a policy π. This helps in making decisions at each state. An optimal policy is to select the actions that have maximum Q-values at that state, thereby maximizing rewards. For State (s), Action (a), Reward (r), and Next State (s′), the optimal Q-value at each state is approximated using the Bellman equation:

Q(s, a) = r + γ maxa' Q(s', a')………. (2)

This equation is defined by the maximum future reward for this state (s). Action (a) is the immediate reward (r) plus a maximum future reward for the following state (s′). γ is the discount factor that determines how much weight should be given to future rewards while calculating Q-function for the state. The main idea in Q-learning is that we can iteratively approximate the Q-function using Bellman's equation.

The basic Q-learning algorithm is as follows:

• Initialize Q (numstates, numactions) arbitrarily.

• Observe initial state s.

• Repeat.

• Select and carry out an action a.

• Observe reward r and new state s'.

• Q[s, a] = r + γ(maxa' Q[s', a']).

• s = s', until terminated.

The Q-values of states and actions are initially random, and they converge after several iterations. The Q-value is updated accurately in each iteration. We can use the concept of Q-learning by using a deep neural network and the experience replay algorithm, which uses an agent's experience to update the Q-values. The training of the agents using this algorithm is explained in the next section.

1. **Proposed system of RL with sentiment scores**

The proposed system trains an agent to decide to buy, sell or hold a stock when given a state as input. Along with the consideration of sentiment scores, which are results from the sentiment analysis of news regarding that specific stock on the corresponding date.

The proposed system contains two essential modules:

(a)   **RL module without sentiment scores.** This module decides whether to buy, sell, or hold (do nothing) a stock given a state as input.

(b)   **RL module with sentiment scores.** This module takes into consideration the news of the stock for each day and trains the agent to decide whether to buy, sell, or hold the stock.

***4.1 Input***

The input for the system consists of stock data from Apple which also has compound scores (sentiment scores) on daily basis.

***4.1.1 Stock Data***

The stock data of the company includes the following fields: Open, Close, High, Low, and Volume of the stock. The "High" field refers to the highest price of the stock in the considered period, while the "Low" field refers to the lowest price in the considered period. The "Volume" field refers to the number of shares traded during the given period. The "Open" field refers to the opening price in the considered period, while the "Close" field refers to the closing price during the considered period. The considered period for the stock data is one day. The daily stock data of a company for the last 10 years is used as input. The stock data is pre-processed to identify missing values and to calculate moving averages.

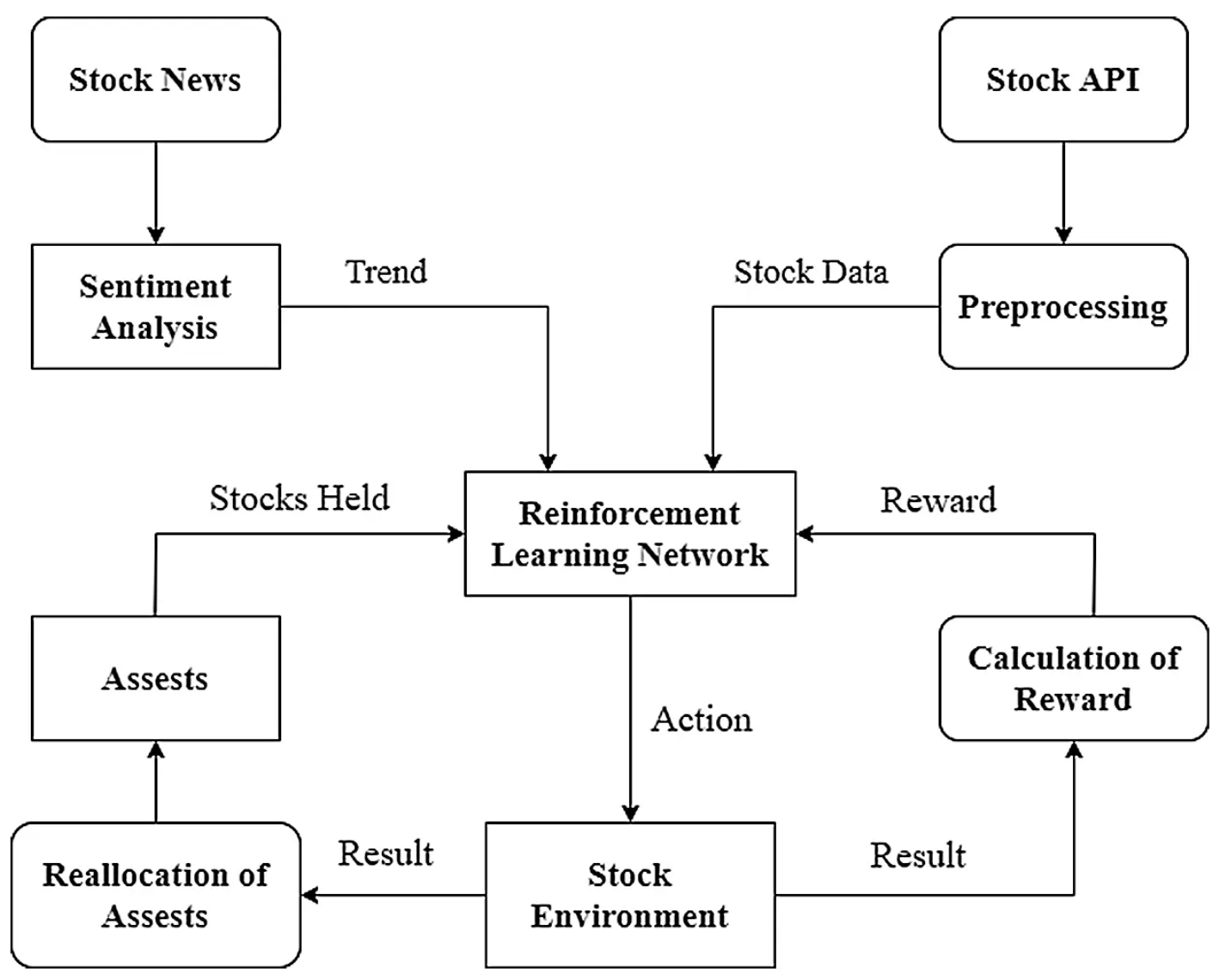


Fig 2: Proposed Algorithms System

* + 1. ***Sentiment Scores***

A positive score indicates how positive the sentiment was on a particular day. A negative score indicates how negative the sentiment was on a particular day. The compound score is a score between −1 (too negative) and +1 (too positive). The compound score is used as an input to the RL module.

1. **RL Network**

To formulate stock trading as an MDP and tackle it with RL, an RN network is constructed, consisting of:

(a)   **Environment:** Stock data of a company.

(b)   **State:** Stock information on a day, along with the sentiment score of the news on that corresponding day.

(c)   **Actions:** Sell, Buy, or Hold.

(d)   **Rewards:** Rewards are feedback signals given back to an agent when that agent takes an action. For each action, a positive reward is given when profit is made, and a negative reward is given when an agent's action leads to loss.

1. **Deep neural network**

The code provided implements a Deep Neural Network (DNN) in the context of reinforcement learning (RL). The DNN is trained to estimate the Q-value function, which is a measure of the expected future reward for taking a particular action in a given state. In this case, the DNN is used to estimate the Q-value function for a stock trading agent.

The DNN architecture used in this code consists of three layers, with the first layer having 64 neurons, the second layer having 32 neurons, and the output layer having three neurons (one for each possible action). The input to the DNN is a state vector consisting of five features: the open price of the stock, the 5-day moving average of the stock price, the current portfolio value, the cash held by the agent, and the number of stocks held by the agent.

The DNN is trained using a variant of Q-learning called Deep Q-Network (DQN) learning. The agent stores its experiences (i.e., the state, action, reward, next state, and done flag) in a memory buffer, and a batch of experiences is randomly sampled from the buffer for training the DNN. During training, the DNN is updated to minimize the difference between the predicted Q-value and the target Q-value. The target Q-value is calculated using the Bellman equation, which takes into account the reward received for the current action and the estimated future reward for the next state.

To improve the stability of the learning process, a target network is used to generate the target Q-values. The weights of the target network are updated periodically (in this case, every 50 episodes) to match the weights of the main network. Additionally, an epsilon-greedy exploration policy is used to balance exploration and exploitation during training. The exploration rate starts high (at 1.0) and decays exponentially over time to encourage the agent to rely more on its learned Q-values as training progresses.

1. **Training of RL Agent**

The methodology for training of a stock market trading agent is given below. The agent starts with initial open cash and several stocks. Batch size is defined. A batch is the set of tuples, which contains State, Action, Reward, and Next Size. It is used in the experience replay function for updating the network. Discount factor γ is also defined.

For each day in the dataset:

1. *The state is defined with stock opening price, 5-day moving average price of the stock, stocks held, cash held, and sentiment score of the previous day.*
2. *Action is taken, giving the state as an input to the deep neural network.*
3. *The reward is calculated according to the action taken using the price change percentage given by:*
4. *Price change percentage = ((Current stock price - Five day avg.) / Five day avg.) x 100………. (Equation 3)*
5. *Portfolio value is the net worth of the investor. It is calculated by:*
6. *Portfolio value = (Stocks held x Current stock price) + Cash held………. (Equation 4)*
7. *Assets (cash held and stocks held) are recalculated according to the action taken, and then the next state is defined.*
8. *Tuple (State, Action, Reward, Next state) is added to the agent memory.*
9. *If the agent memory is greater than the defined batch size, the experience replay function is called with the agent memory as input to update the deep neural network. The experience replay algorithm is given below.*
10. **Experience replay**

Experience replay is a technique used in reinforcement learning to improve the efficiency and stability of the training process. It involves storing the agent's experiences in a memory buffer and randomly sampling batches of experiences from the buffer to train the deep neural network.

The experience replay algorithm involves the following steps:

Define hyperparameters: This step involves setting the hyperparameters for the algorithm, such as the batch size, discount factor (gamma), memory size, exploration rate (epsilon), minimum exploration rate (epsilon\_min), exploration rate decay (epsilon\_decay), learning rate, and update target network frequency (update\_target\_network).

Define the deep neural network model: This step involves defining the structure of the deep neural network that will be used to predict the Q values for each state-action pair. In the example code, a sequential model with three dense layers is used, where the input layer has 5 neurons (representing the state), the first hidden layer has 64 neurons, the second hidden layer has 32 neurons, and the output layer has 3 neurons (representing the Q values for each action).

Define the target network model: This step involves creating a copy of the deep neural network model that will be used as the target network for calculating the target Q values. The weights of the target network are initially set to the weights of the deep neural network.

Define the experience replay function: This step involves defining the experience replay function that will be used to train the deep neural network using batches of experiences sampled from the memory buffer. The function takes the agent memory as input and performs the following steps:

Sample a batch of experiences from the memory buffer.

Unpack the batch into states, actions, rewards, and next states.

Predict the Q values for the current states and the next states using the deep neural network and the target network, respectively.

Calculate the target Q values using the Bellman equation:

Q(s,a) = r + gamma \* max(Q(s',a')).

If the next state is terminal, the target Q value is simply the reward.

Update the Q values for the current states and actions using the calculated target Q values.

Steps involved in experience replay algorithm:

1. *Initialize the agent memory: This step involves creating an empty memory buffer to store the agent's experiences.*
2. *Iterate through each episode: This step involves iterating through each episode of the training process and performing the following steps:*
3. *Initialize the portfolio value, cash, and stocks held.*
4. *Iterate through each day in the dataset and perform the following steps:*
5. *Define the state for the current day.*
6. *Take an action using an epsilon-greedy policy.*
7. *Calculate the reward for the action taken.*
8. *Update the portfolio value with the current cash and stock holdings.*
9. *Update the agent memory with the current state, action, reward, next state, and done flag.*
10. *Perform experience replay if the memory buffer is full.*
11. *Update the target network if the update target network frequency has been reached.*

The equations used in the experience replay algorithm are as follows:

Bellman equation: Q(s,a) = r + γ \* max(Q(s',a'))

where s is the current state, a is the action taken, r is the reward received, gamma is the discount factor, s' is the next state, and max(Q(s',a')) is the maximum Q value for the next state.

A picture containing diagram

Description automatically generated

Fig 3: Reinforcement Learning Module

1. **Results and Discussion**

In this section, I have created sub-sections to discuss all the inputs and methods used to obtain the desired results.

***9.1 Dataset***

The initial dataset contains the stock information on each market day for Apple from 2006 to 2016. Each day stock information contains Date, Open Price (price of stock at the start of that day), Close Price (price of stock at the end of that day), High Price (highest price of stock in that day), Low Price (Lowest price of stock on that day), Adjusted Close Price, Volume (Number of stocks traded on that day), Compound Scores (Sentiment Scores)

The open price of Apple stock plotted against each market day is shown in Fig 4 . The X-axis represents the year, and the Y-axis represents the price in dollars. It is seen that there are rise and fall in stock prices across the years.

Chart, line chart

Description automatically generated

Fig 4: Open price curve of APPL

***9.2 Reinforcement Learning (RL) on Dataset***

The dataset is split into training and testing sets. The training dataset consists of 1513 rows and the testing dataset consists of 1000 rows. Two scenarios are considered for comparison with the proposed system.

* 1. RL without sentiment input.
  2. RL with sentiment input.

These scenarios are compared in performance with the proposed system using a portfolio value at each time. The portfolio value is the investor's net value of assets, which is given by adding cash and the market value of stocks held by the investor.

***9.2.1 Training with RL Without Sentiment score Input***

The parameters used for training are:

1. Batch size = 50
2. gamma = 0.95
3. memory size = 10000
4. epsilon = 1.0
5. epsilon minimum = 0.01
6. epsilon decay = 0.995
7. learning rate = 0.001
8. num episodes = 50
9. State = (open price, five-day moving average price, portfolio value, cash held, stocks held).

Training is performed on a training dataset with the given training parameters. All rows in the dataset are iterated for training for 50 episodes. Batch size is the number of tuples containing the State, Action, Reward, and Next State used for updating weights and biases of the model. The model is saved after training for the testing purpose.

***9.2.2 Testing with RL Without Sentiment score Input***

The model generated from the training phase is used in the testing phase. The model starts with initial cash and buys a fixed number of stocks using that cash as in the benchmark model testing. The initial cash is $10000, and the number of stocks brought initially is 25. The remaining cash after buying the stocks on the starting day is calculated. The graph for the portfolio values is shown below.

The X-axis of the graph represents the days, and the Y-axis represents the portfolio value in dollars.

Chart

Description automatically generated

Fig 5: Portfolio curve without sentiment scores

The above graph shows how the portfolio value fluctuates over each day. The trained model is used on test data which allows agent to buy, sell or hold the stocks. After the end of the test data, the final portfolio is $14260.097783969999. Which clearly shows a profit of $4260.

***9.2.3 Training with RL with Sentiment Input***

The parameters used for training are:

1. Batch size = 50
2. gamma = 0.95
3. memory size = 10000
4. epsilon = 1.0
5. epsilon minimum = 0.01
6. epsilon decay = 0.995
7. learning rate = 0.001
8. num episodes = 50
9. State = (open price, five-day moving average price, portfolio value, cash held, stocks held, sentiment score).

Training is performed on a training dataset with the given training parameters. All rows in the dataset are iterated for training for 50 episodes. Batch size is the number of tuples containing State, Action, Reward, and Next State used for updating weights and biases of the model. The model is saved after training for testing purposes.

***9.2.4 Testing with RL with Sentiment Input***

The model generated from the training phase is used in the testing phase. The model starts with an initial cash and a fixed number of stocks. The portfolio value on each day is calculated. A graph for portfolio values is plotted and compared with that of without sentiment scores. The comparison of trading with the RL model without sentiment and the RL model with sentiment is illustrated in the below figures. The X-axis of the graph represents the days and the Y-axis represents the portfolio value in dollars.

Chart

Description automatically generated

Fig 6: Portfolio curve with sentiment scores

The above graph shows how the portfolio value fluctuates over each day. The trained model is used on test data which allows the agent to buy, sell or hold the stocks. After the end of the test data, the final portfolio is $16892.91943635. Which clearly shows a profit of $6892.

1. **Comparison between with and without sentiment scores:**

Chart, scatter chart

Description automatically generated

Fig 6: Portfolio curve without and with sentiment scores

***10.1 Portfolio results from testing data comparisons:***

Initial Portfolio without sentiment scores value is: 10000

Final Portfolio value without sentiment scores is: 14260.097783969999

Profit without sentiment scores is: 4260.097783969999

Initial Portfolio with sentiment scores value is: 10000

Final Portfolio value with sentiment scores is: 16892.91943635

Profit with sentiment scores is: 6892.919436349999

I have found that training the model along with sentiment scores leads to better results. This indicates that sentiment plays an important role in the stock market and has an impact on traders. I provide the reinforcement learning model that utilizes sentiment scores obtained from daily news related to specific stocks.

1. **Conclusions and Future Works**

In conclusion, predicting stock prices and forecasting market conditions is a difficult task that has been studied extensively. Naive approaches, deep learning techniques, and RL techniques have been explored for stock market forecasting. RL, although efficient for making automated decisions, requires high-end processors for real-time performance.

The proposed system presented in this study incorporates RL and market sentiment for decision-making, and the results show that including sentiment analysis leads to better decision-making compared to trading based on statistics alone. However, this system currently only works for a single stock, and future work should involve extending it to multiple stocks.

To extend the proposed system to multiple stocks, news for different stocks would need to be segregated and sentiment analysis should be implemented in a distributed manner. A major challenge to be addressed is detecting fake news for more efficient performance. In addition, the proposed system could be further improved

by implementing real-time sentiment analysis of news for intra-day trading.

1. **Appendix**

* The code for the project is available on GitHub.

<https://github.com/mkundan1489/CMSE890-001_project>

* Dataset:

<https://www.kaggle.com/datasets/lorilaz/apple-news-headline-sentiment-and-stock-info?select=aaplCombined.csv>