Algorithmic Trading using Sentiment Analysis and Reinforcement Learning

Simerjot Kaur (SUNetID: sk3391 and TeamID: 035)

Abstract

This work presents a novel algorithmic trading system based on reinforcement learning. We formulate the trading problem as a Markov Decision Process (MDP). The formulated MDP is solved using Q-Learning. To improve the performance of Q-Learning, we augment MDP states with an estimate of current market/asset trend information and sentiment analysis of news articles of each asset which are estimated using Neural Networks. Our proposed algorithm is able to achieve a Sharpe Ratio of 2.4 when we invest $10,000 as compared to a baseline Sharpe Ratio of -0.2 and an estimated oracle Sharpe Ratio of 3.0.

1. Introduction

Algorithmic trading also called as automated trading is the process of using computers programmed to follow a defined set of instructions for placing a trade in order to generate profits at a speed and frequency that is impossible for a human trader. An important factor affecting the trading decisions is the ability of predicting stock market movement. The prediction of stock markets movement is considered to be a challenging task of financial time series prediction due to the complexity of the stock market with its noisy and volatile environment, considering the strong connection to numerous stochastic factors such as political events, newspapers as well as quarterly and annual reports.

In this project, I want to explore the possibility of building a machine learning agent that tries to learn an optimal trading policy/strategy using several machine learning techniques like reinforcement learning. The problem we are trying to solve in this project can be summarized as: ***"Train a ML Agent to learn an optimal trading strategy based on historical data and stock market news in order to maximize the generated profits."***

In particular, the project would involve the following sub-steps, as shown in Figure 1 below:

1. Formulate the trading problem as a Markov Decision Process (MDP) which is then solved using Q-learning with functional approximations. The states of the MDP not only involve historical price, number of stocks, cash in hand but also trend information (step 2) and sentiment score (step 3).
2. Perform Trend Analysis using technical market indicators as input to neural networks and obtain stock market trend as the output signal.
3. Perform Sentiment Analysis on news articles of each asset.

Rather than maximizing profits, most modern fund managers attempt to maximize risk-adjusted return as advocated by Modern Portfolio Theory. The Sharpe Ratio is the most widely-used measure of risk-adjusted return (Sharpe) and can be calculated as the average of the profit/returns generated at time *t* normalized over the standard deviation of the profit/returns generated at time *t*. Thus for this trading system, the performance criterion will be based on Sharpe Ratio. The oracle [1] in terms of Sharpe Ratio is that usually, any Sharpe ratio greater than 1 is considered acceptable to good by investors. A ratio higher than 2 is rated as very good, and a ratio of 3 or higher is considered excellent.

In our first approach, described in Section 4, we formulated the MDP consisting of historical price, number of stocks, cash in hand as states and solved using reinforcement learning. In section 5, we extend our approach by augmenting the states with current market/industry trend information and then solve the MDP using reinforcement learning. In Section 6, we further extend our approach by augmenting the states with Sentiment Analysis performed on news articles. Section 7 provides a summary and comparison of our results. Section 8 further identifies the current limitations of our model. Finally, section 9 summarizes and concludes this paper and lays foundation for our future work.

2. Literature Review

Various different techniques have been implemented in literature to train ML agents to do automated trading using machine learning techniques. For instance, *[Pierpaolo G. Necchi, 2016][2], [David W. Lu, 2017][3], [Xin Du, Jinjian Zhai, Koupin Lv, 2009][4], [Jae Won Lee, Jangmin O][5]* all describe different machine learning techniques like deep q-learning, recurrent reinforcement learning, etc to perform algorithmic trading. *[James Cumming, 2015][6]* also wrote a book on the use of different reinforcement learning techniques within the Algorithmic Trading Domain. The major strength of these researches is that they are trying to investigate the best possible learning algorithm so that automated trading can be performed with minimum human intervention.

On the other hand, a large number of implementations have been done to perform sentiment analysis on news articles and twitter feeds. The paper of *[Gyozo Gidofalvi, 2011][7]* predicts the short-term stock price movements using financial news articles. The paper of *[Naveed Ahmad, Aram Zinzalian, 2010][8]* explores stock volatility forecasting from quarterly earnings call transcripts of the 30 DOW component stocks. The paper of *[Qicheng Ma, 2008][9]* presents a scheme in which 29 stock names are picked from DJI and related articles and are classified as positive or negative using Naive Bayes and Maximum Entropy classifier. The paper *[Ravi Parikh, Matin Movassate, 2009][10]* uses Unigram and Multi-nomial Bigram Naive Bayes to detect sentiment. The paper of *[Gabriel Fung, Jeffrey Yu, Hongjun Lu,*

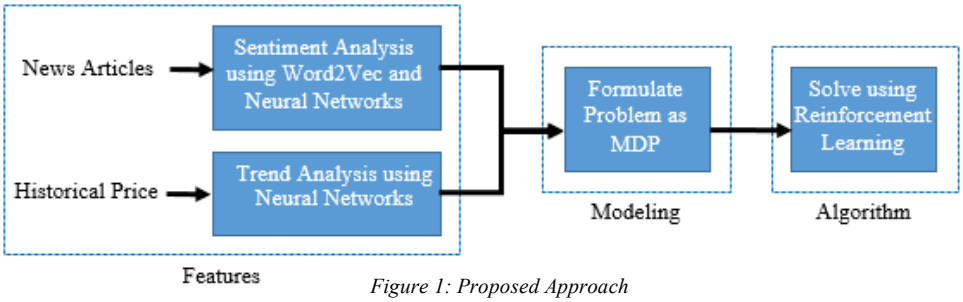

Figure 1: Proposed Approach

2005][11] predicts the movements of stock prices based on the contents of the news stories. All these researches are trying to classify the news more accurately, thereby trying to obtain significant accuracy.

In this project we follow a novel approach of combining both the implementations where we gather information from various sources that have big impact on the stock market and supplying this additional information to better learn an optimal policy/strategy. Thus the ML agent not only just learns an optimal trading strategy based on historical prices but also on additional information based on sentiment and trend of the market to make an informed decision.

3. Dataset Used

5-years of daily historical closing prices (2011-2016) for various stocks were obtained from Yahoo Finance[12] to form the dataset. For this project, the results are restricted to a portfolio consisting of 2 stocks *[Qualcomm (QCOM), Microsoft (MSFT)]*.

10-years of news articles have also been obtained from Reuters Key Development Corpus[13] for both Qualcomm and Microsoft. 5-years of these news articles (2006-2010) are used as training dataset and the remaining (2011-2016) are used as test dataset to perform Sentiment Analysis. The headlines were manually classified to obtain the ground truth sentiment score: +1 if the news provides a positive sentiment and -1 if the news provides a negative sentiment.

4. Problem Formulation

This section describes how the problem of choosing when to BUY/HOLD/SELL a portfolio of stocks is formulated as a Markov Decision Process (MDP). The section further elaborates on how the MDP is solved by learning an optimal policy using Q-learning with functional approximations. It may be noted that since the state space with real valued stock prices is really huge, the state space has been discretized for the stock prices.

MDP Formulation:

MDP[16] can be formulated by describing its States, Actions, Transition Probability, Rewards, and discount factor.

States: *[(#of Stocks for each asset), (Current Stock Price for each asset), Cash in Hand]*

The first part of the state consists of a tuple containing number of stocks for each asset. The second part of the state consists of a tuple containing the daily closing stock price for each asset. Finally, the third part of the state consists of cash in hand which is evaluated at every time step based on the action performed.

Initial State: *[(0, 0...), (S1, S2...), $10,000]* i.e. the agent has 0 stocks for each asset and only $10,000 as an initial investment.

Actions: At any point the agent chooses from three actions: *BUY, SELL, and HOLD*. Action BUY buys as many stocks for each asset as possible based on the current stock price and cash in hand. Action SELL sells all the stocks in portfolio and adds the generated cash to cash in hand. Action HOLD, does nothing, i.e. neither sells nor buys any stock of an asset.

Transition Probability: The transition probability is chosen to be *1* always as whenever the action is BUY/SELL we are sure to BUY/SELL the stocks of an asset. Here the randomness in the system comes from the fact that the stock price changes just after a stock is bought or sold i.e. after every time step.

Rewards: The reward at any point is calculated as the *[Current Value of the Portfolio - Initial Investment]*.

Discount Factor: In this project, the discount factor is assumed to be *1* always.

Solving the MDP:

The MDP described above was solved using vanilla Q-Learning algorithm with functional approximations[17].

The algorithm can be described as:

On each (s, a, r, s′):

$$\widehat{Q_{opt}}(s,a) \leftarrow (1-\eta)\widehat{Q_{opt}}(s,a) + \eta(r + \gamma \widehat{V_{opt}}(s'))$$

where: $\widehat{V_{opt}}(s') = \max_{a' \in Actions(s')} \widehat{Q_{opt}}(s',a')$

s = current state, a = action being taken, s' = next state, γ = discount factor, r = reward, η = exploration probability.

As Q-learning doesn't generalize to unseen states/actions, function approximation has been used which parameterizes $\widehat{Q_{opt}}$ by a weight vector and feature vector which can be described as:

On each (s, a, r, s′):

$$w \leftarrow w - \eta\left[\widehat{Q_{opt}}(s,a;w) - \left(r + \gamma \widehat{V_{opt}}(s')\right)\right]\phi(s,a)$$

where: $\widehat{Q_{opt}}(s,a;w)$ *= prediction and* $r + \gamma \widehat{V_{opt}}(s')$ *= target*

For our problem, we used the following feature vectors: (a) Number of Stocks of each asset, (b) Current Stock Price of each asset and, (c) Cash in Hand.

Also, in order to deploy the tradeoff between exploration and exploitation, epsilon-greedy algorithm has been used which explores with probability ε and exploits with probability 1−ε. An exploration probability of 0.2 has been chosen for this project.

Results:

This section discusses the performance of the above implemented Q-learning system. The Q-learning system was run on a dataset containing 5-year stock prices and number of trials as 10,000.

The plot below shows how the Sharpe Ratio evolves with increasing historical time period. As can be observed, the ML agent slowly but steadily learns an optimal strategy that increases the Sharpe Ratio as more historical data is provided, thus displaying incremental performance and is successful in achieving the Sharpe Ratio of 0.85.

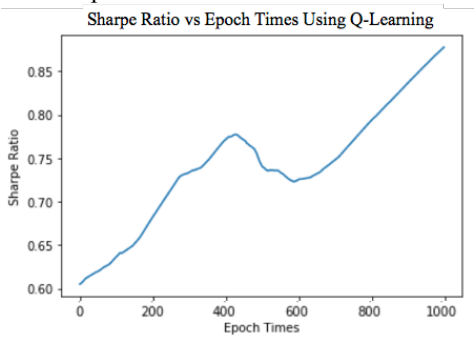

Figure 2: Sharpe Ratio vs historical time period using Q-Learning

To validate that the algorithm was effective in learning the optimal strategy, Monte-Carlo simulation has been performed as baseline. In this simulation, the agent is forced to choose an action at random. As can be observed in the plot below, the ML agent generates negative Sharpe Ratio, hence validating the Q-learning algorithm for this problem.

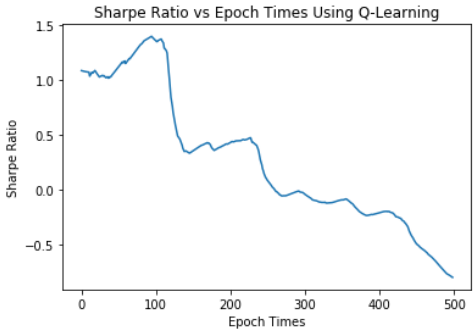

Figure 3: Sharpe Ratio vs historical time period using MCS

5. Problem Formulation Augmented with Trend Analysis

When a human trader makes a trading decision, in addition to the stock price, his decision is also based on his view of the market/industry's trend so that he is able to profit from the upward movement and avoid the downfall of a stock. In this section, we are trying to capture the market's current trend and use this trend information as an additional feature in the state definition of our MDP formulation. The practical implementation of the above approach entails two steps: (A) Finding the current trend information (B) Augmenting the trend information into MDP formulation.

5.A Using Market Technical Indicators and Neural Network to find Trend Information

To identify the trend information, six popular market technical indicators[15] calculated from the historical stock price are used as the input features for the neural network. The output of this neural network is used to perform the classification into two states, 1 if an upward trend is observed and 0 if downward trend is observed.

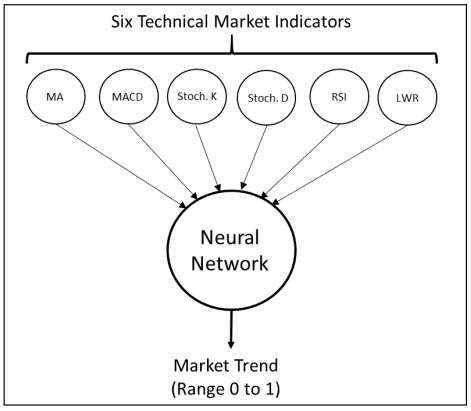

Figure 4: Trend Analysis performed using neural networks

The six technical indicators that are used as input for the neural network can be calculated as shown below:

1. Simple Moving Average: t=15

$$MA_t = \frac{1}{t}\sum_{i=1}^{t} cp(i)$$

2. Moving Average Convergence and Divergence

$$MACD = EMA_{12} - EMA_{26}$$

$$where\ EMA(i) = \big(cp(i) - EMA(i-1)\big) * \left(\frac{2}{t+1}\right) + EMA(i-1)$$

3. Stochastic K

$$K\%(i) = \frac{cp(i) - L_t}{H_t - L_t} * 100$$

4. Stochastic D

$$D\%(i) = \frac{K\%(i-2) + K\%(i-1) + K\%(i)}{3}$$

5. Relative Strength Index

$$RSI = 100 - \left(\frac{100}{1+RS}\right)$$

$$where\ RS = \frac{Average\ of\ t\ day's\ up\ closes}{Average\ of\ t\ days'sdown\ closes}$$

6. Larry William's R%

$$R\%(i) = \frac{H_t - cp(i)}{H_t - L_t} * 100$$

Where cp(i) is the closing price, L_t is the lowest price of last t days, H_t is the highest price of last t days. Also, the six technical indicator values represent continuous values in different ranges. So the input data is scaled in the range 0–1 using the min max normalization as follows:

$$y = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Where y = normalized value, x = value , x_{min} = minimum value and x_{max} = maximum value of the series to be normalized.

The scaled technical indicators are then inputted into the neural network. An inbuilt neural network toolbox in python, HP-ELM[14], has been used in this project to get the output trading signal (OTS) in range 0-1. The uptrend and down trend are classified using the following methodology:

If OTS > Mean (OTS), predicted trend is up (1) else down (0).

The chart below shows historical price (blue) and predicted trend states(orange) (~2 years) by the neural network for test data (QCOM stock).

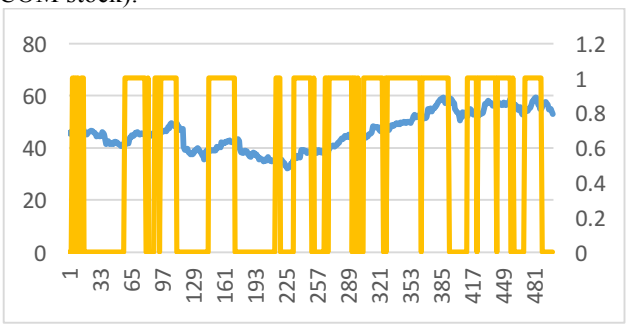

Figure 5: Trend States vs Historical Price for QCOM stock

5.B Augmenting the trend information into MDP formulation

The above section described how we can identify current trend in stock data using neural networks. To provide this information to our ML agent, we augmented the state of the system with this information and the new state implemented was [(#of Stocks for each asset), (Current Stock Price for each asset), Cash in Hand, (trend of each asset)]. All other aspects of the MDP definition are same as before.

The MDP was still solved using Q-learning approach as in section 4, with the only difference being that the trend of asset is also being used as feature in our function approximation based Q-Learning.

Similar to Section 4, MDP was run over the data-set consisting of 5-year time period and 10,000 trials and Sharpe Ratio was reported for each data point. The plot below shows that the Sharpe Ratio of 1.4 is obtained and is much higher and stable as compared to Section 4.

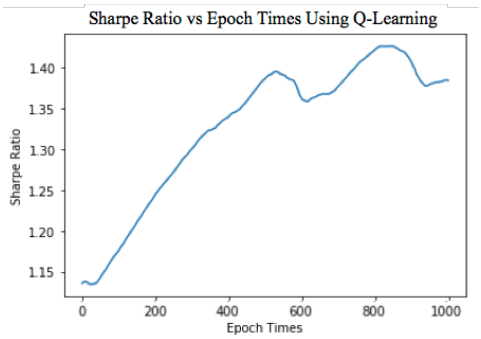

Figure 6: Sharpe Ratio vs historical time period using Q-Learning

Similar to section 4, Monte-Carlo Simulation was considered as a baseline. The plot below shows that providing the trend information also does not help in improving the performance obtained from random actions and Monte-Carlo Simulation.

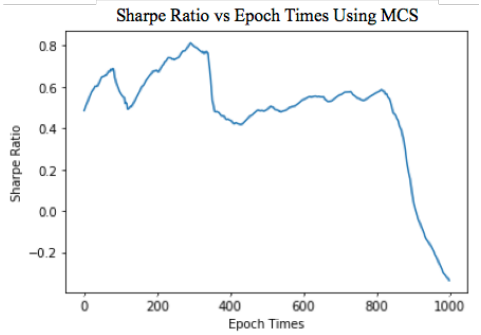

Figure 7: Sharpe Ratio vs historical time period using MCS

6. Problem Formulation Augmented with Sentiment Analysis and Trend Information

Due to the volatility of the stock market, price fluctuations based on sentiment and news reports are common. As humans, we can easily interpret the sentiment of a news article. However, the problem lies when we have multiple stocks in our portfolio, most of which are being mentioned frequently in the news or social media sites. The average adult reads 300 wpm and takes approximately 2 minutes to read an article with 600 words. Within 2 minutes, 10 of the portfolio stocks generate over 100 articles. It is impossible for humans to keep up with the massive data overload which is where sentiment analysis comes into play.

In this section, sentiment analysis is being performed on the news articles of each stock and use this sentiment score as an additional feature in the state definition. The practical implementation of the above approach entails two steps: (A) Perform sentiment analysis (B) Augmenting the sentiment score into MDP formulation.

6.A Sentiment Analysis using News Articles of each asset

The sentiment analysis has been performed in two steps, as shown in the figure 8 below. The first step involves passing the news articles through a pre-trained Word2Vec model and obtaining N-Dimensional Vector for each headline. The second step involves passing these vectors as inputs to a neural network and obtaining the classification also known as sentiment score.

Word2Vec Model:

As a first step, we used a pre-trained Word2Vec Model for Reuters News Corpus (using NLTK Library[18]). Word2Vec[19] is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

We use the pre-trained Word2Vec model to obtain a 100-dimensional vector for each of the headline obtained for Qualcomm and Microsoft (2006-2016) from Reuters Key Development Corpus.

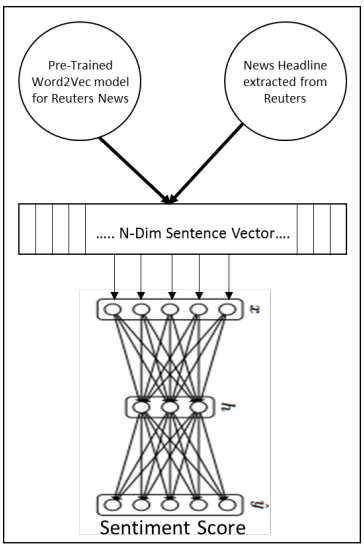

Figure 8: Sentiment Analysis using Word2Vec Model and Neural Networks

Neural Network:

Subsequently, we use these 100-dimensional vectors as input to a neural network with a single hidden layer (120 dimensions) and cross entropy loss. The sigmoid function has been used as activation for the hidden layer, and softmax function for the output layer. The forward-propagation and back-propagation[20] for the cross entropy loss with regularization (λ=0.0001) are implemented to perform the classification.

The model was trained on news from year 2006 to 2010. The news headlines from year 2011 to 2016 acted as a test set. The figure 9 below shows the training and test set accuracy using the proposed method. As is evident, the model is able to achieve a training accuracy of ~100% with the test set accuracy being ~80%. It may be noted that finding the sentiment from just the headline is rather a difficult task as headlines can often be ambiguous.

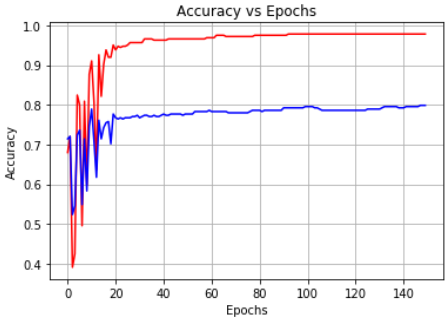

Figure 9: Accuracy of training(red) and test(blue) set vs the number of epochs

6.B Augmenting the sentiment score into MDP formulation

The above section described how we can obtain the sentiment score using Word2Vec model and neural networks. Consequently, we augmented the state of the system with this information and the new state implemented was [(# of Stocks for each asset), (Current Stock Price for each asset), Cash in Hand, (trend of each asset), (sentiment score of each asset)]. All other aspects of the MDP definition are same as before.

The MDP was still solved using Q-learning approach as in section 4 and 5, with the only difference being that the trend of asset and sentiment scores are also being used as feature in our function approximation based Q-Learning.

Similar to Section 4 and 5, MDP was run over the data-set consisting of 5-year time period and 10,000 trials and Sharpe Ratio was reported for each data point. The plot below shows how the Sharpe Ratio evolves with increasing historical time period. It may be noted that the Sharpe Ratio of 2.4 is obtained and is much higher and stable as compared to Section 5.

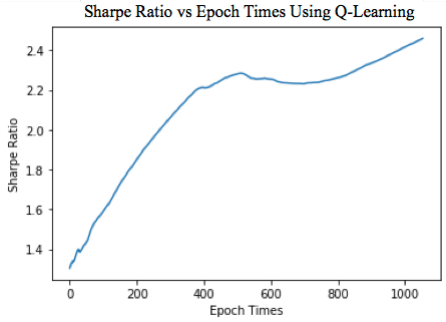

Figure 10: Sharpe Ratio vs historical time period using Q-Learning

Similar to previous sections, Monte-Carlo Simulation is considered as a baseline. The plot below shows that providing the trend and sentiment information also does not help in improving the performance obtained from random actions and MCS.

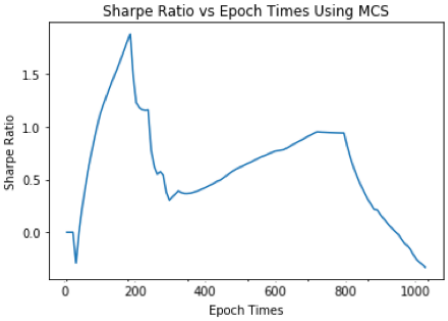

Figure 11: Sharpe Ratio vs historical time period using MCS

7. Summary of Results

The table below summarizes the Sharpe Ratio achieved from 10,000 trials using the various proposed algorithms.

Methods		Sharpe Ratio
RL	**Q-Learning**	0.85
	MCS	-0.5
RL + Trend	**Q-Learning**	1.4
	MCS	-0.2
RL + Trend + Sentiment Score	**Q-Learning**	2.4
	MCS	-0.2

Table 1: Summary of Results from various proposed algorithms

While all algorithms are better than our baseline Monte-Carlo simulations (MCS), as expected, the best performance is achieved when we use Q-Learning to solve MDP augmented with trend and sentiment analysis. This is expected because adding the trend information and sentiment scores to our state helps the ML agent to make much more informed decisions.

8. Challenges and Error Analysis

We observed that while our proposed approach works well in a lot of scenarios, there are still significant challenges that need to be solved to build high performing autonomous trading system. Some of the major challenges/limitations in our current approach are:

a. **Large State Space:** The exponential increase in the state-action pair space as the number of assets in our portfolio increases. Since our state definition consists of *(number of stocks for each asset, stock prices of each asset, cash in hand, trend path followed by each asset, sentiment score of each asset)*, as the number of assets increase, the action space increases exponentially. This increases the run-time of our system considerably, and limits the performance of the system.

b. **Accuracy Improvement of Sentiment Analysis:** The accuracy of our proposed method of estimation of sentiment score can be improved by taking the content of news articles rather than just taking the headlines as headlines can often be ambiguous.

9. Conclusion and Future Work

In this paper, we implemented reinforcement learning algorithm particularly Q-Learning with functional approximations, to train a ML agent to develop an optimal strategy to perform automated trading. In the first section, the ML agent was successful in learning the optimal policy, achieving a Sharpe Ratio of 0.85. In order to improve the learning efficiency of the agent, Trend Analysis using Neural Networks was performed and the MDP was augmented with trend information. The agent performed better and was successful in achieving the Sharpe Ratio of 1.4. Furthermore, in order to capture the volatility in the market due to news reports, Sentiment Analysis using Word2Vec model and Neural Networks was performed. The sentiment score was then augmented to the MDP formulation and solved. The agent performed significantly better and achieved a Sharpe Ratio of 2.4.

However, although the average profit generated in 10,000 trials is closer to Oracle (Sharpe Ratio = 3.0 being excellent), we still see a significant run-to-run variation. The run-to-run variations could be because of limited data set, i.e. we only have stock data and news articles test data for 5-year time period.

Finally, our current work can be extended in various ways. For instance, while performing Sentiment Analysis, only the news headlines have been used. This can be further extended to include the content of news article which will help the model to classify the news more accurately, thereby increasing the accuracy. Further, given the large state space, it might be worthwhile to try to use policy gradient methods which directly try to estimate the best policy without really generating the Q-Value. Deep Q-Learning and Deep Reinforcement learning methods have recently been applied to many different problems and have achieved very good performance. Based on the initial results from this report, we feel it could be a good idea to try to implement Deep-Reinforcement learning methods for this problem too.

10. References

[1] https://www.investopedia.com/ask/answers/010815/what-good-sharpe-ratio.asp

[2] Pierpaolo G. Necchi, Reinforcement Learning For Automated Trading, 2016

[3] David W. Lu, Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks, July 2017.

[4] Xin Du, Jinjian Zhai, Koupin Lv, Algorithm Trading using Q - Learning and Recurrent Reinforcement Learning, Dec 2009

[5] Jae Won Lee, Jangmin O, A Multiagent Approach to Q-Learning for Daily Stock Trading, Nov 2007

[6] James Cumming, An Investigation into the Use of Reinforcement Learning Techniques within the Algorithmic Trading Domain, June 2018

[7] Győző Gidófalvi, Using News Articles to Predict Stock Price Movements, University of California, San Diego La Jolla, CA 92037, 2001

[8] Naveed Ahmad, Aram Zinzalian, Predicting Stock Volatility from Quarterly Earnings Calls and Transcript Summaries using Text Regression, June 2010

[9] Qicheng Ma, Stock Price Prediction Using News Articles, CS224N Final Report, 2008

[10] Ravi Parikh, Matin Movassate, Sentiment Analysis of User-Generated Twitter Updates using Various Classification Techniques, June 2009

[11] Gabriel Fung, Jeffrey Yu, Hongjun Lu, The Predicting Power of Textual Information on Financial Markets, IEEE Intelligent Informatics Bulletin, Vol. 5. No. 1, June 2005.

[12] Yahoo Finance, https://finance.yahoo.com/

[13] Reuters Key Development Corpus, https://www.reuters.com/finance/stocks/QCOM.O/key-developments

[14] http://hpelm.readthedocs.io/en/latest/index.html, HP-ELM

[15] Rajashree Dash and Pradipta Kishore Dash, A hybrid stock trading framework integrating technical analysis with machine learning techniques, March 2016.

[16] Markov Decision Process, Lecture Notes

[17] Q-Learning and Epsilon Greedy Algorithm, Lecture Notes

[18] NLTK Library, http://www.nltk.org/

[19] Word2Vec Model, https://en.wikipedia.org/wiki/Word2vec

[20] Forward-propagation and back-propagation, Lecture Notes