STONKS - Stock Market Forecasting Using News Articles With Reinforcement And Sequence Learning

Ruslan Sirazhetdinov

May 2023

Abstract

I believe, that there is a correlation between price change direction of a particular share and news articles. The purpose of this project is to predict market price for next day, each time depending on news of the current. The project consists of my own StocksNews dataset, the baseline solution, experiments with existing related solutions, implementation of custom Stocks reinforcement learning environment, making of proposed solution using a combination of reinforcement learning and sequence learning techniques. `https://github.com/irusland/nlpystocks`.

1 Introduction

Stock market if a highly variable environment and for forecasting you need to know how to act in certain situation. There is no doubt that stocks market prediction has highly non-linear relation to the news articles. The situation on the market can change drastically in a moments without any prior information because of the insiders policies and non disclosure agreements. But still, there are many cases when a random tweet for example, from Elon Musk could literally heat up the whole investors community.

The news could be a pretty helpful tool for decision making. It most likely has a little average impact on the market in general, but news could provide some hints for enhancing the forecasts.

It's important to know these aspects during making a decision to invest or not. This work could be a possible baseline for a more complex trading bot that processes all the big data information in realtime.

Primarily, the approach to forecast the timeseries is to train a particular model, it may be a linear or statistically or neural network based. Ensembles or a combination of models is also possible. You have some training procedure, then comes validation on the data that has not been seen by the net.

The difference in my approach is that I have tried to use a human-like actions. The requirements for the model is to make a decision to invest or not,

or temporarily suspend any trading. It's possible with the use of reinforcement learning and building a deep Q-networks. This differs from the regression solving models, because after getting the model's predictions there are many ways to operate with received data. The naive approach could be hard-coding some instructions based on conditional logic. For example: if the market price goes up and reaches the certain threshold we make a decision to buy shares. This is a major flaw of such algorithms in my opinion, and that is why I have chose to mimic real human's actions.

In this work I proposed a sequence learning network architecture and implemented reinforcement environment for training.

1.1 Team

Sirazhetdinov Ruslan sirazhetdinov.rr@phystech.edu

I was the only one participant in the team and responsible for several goals:

1. determination of current state of the art solutions and related works.
2. researching the topic of existing datasets that will suit the need of this work.
3. preparation of news and stocks dataset, with custom human-like crawler.
4. making a simple baseline solution.
5. adapting existing models from related works for dataset.
6. developing of custom reinforcement learning environment.
7. implementation of proposed solution.
8. preparing a source code repository.
9. gathering the results.
10. writing a report.

2 Related Work

In this section I will provide my understanding and implementation of currently available solutions to the forecasting problem.

2.1 Sequence learning using recurrent logic

A Robust Predictive Model was presented in the paper [Sen and Mehtab, 2021]. The key idea for the prediction is to process not only the current moment of time but also previous data with some kind of lag measured in days. The authors introduce us to their successful experiment with recurrent neural network. They

also show the results of some linear-based models but, they state that "stock price movement is a highly non-linear process" with which I agree.

The experiment consists of preparing the data with some kind of Natural Language Processing, which was unclear to me, how did they manage to get embeddings from sentences. So I chose the TF-IDF embeddings, and min-max scaler for gathered data.

The main model contained LSTM [Hochreiter and Schmidhuber, 1997] module and fully connected layers to predict a single float number - the price. LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN). The main advantage of LSTMs over traditional RNNs is their ability to learn and store long-term dependencies in the input sequence. This is achieved through the use of specific memory cells that store information and a set of gates that control access to these memory cells. The memory cells and gates allow LSTMs to selectively remember or forget information from the input sequence, which makes them good for tasks that rely on understanding context and long-term dependencies.

I was not able to find a suitable source code for solution, some haven't provided a source code links, and others have been implemented in different programming languages like java, Since I am using python as a tool in this research, I have tried to implement my own representation according to experiment notes in papers.

The news were classified into moods such as positive, negative, neutral and then one-hot encoded. The Lag was set to 3, as the best one mentioned in paper so model receives news from 3 previous time points. The hidden size was 42, this parameter controls the number of parameters in the LSTM layer, and thus affects the model's capacity to capture complex patterns and dependencies in the input data.

After implementation and 100 epochs of MSELoss via Adam optimisation algorithm I achieved pretty decent results which will be described in Results6

2.2 Sequence learning using Fuzzy logic

A Self-Organizing Fuzzy Neural Network is proposed by [Salimi-Badr and Ebadzadeh, 2022]. SOFNN is a type of artificial neural network that combines the techniques of fuzzy logic and neural networks. Interestingly enough, this architecture solves the task of unsupervised learning. It's designed to automatically identify the underlying structure of the data and create fuzzy rules that accurately describe the input-output relationship of the system being modeled. It consists of four layers: input layer, fuzzy layer, normalization layer, and output layer.

The learning process in SOFNN involves the self-organization of the fuzzy rules, which are generated through a clustering that groups similar data points together and determines the centers of the fuzzy sets. These fuzzy rules are then optimized to minimize the error between the network output and the desired output.

The SOFNN was used in work [Bollen et al., 2011] and author proposed that "Twitter mood predicts the stock market". And tweets were segreagated into

multiple clusters and then with a combination of market data passed into lower networks.

While implementing an architecture, for I chose KMeans clusterization algorithm [Hartigan and Wong, 1979] for sequence locator, the module which is responsible for finding the best matching fuzzy rule for a given input pattern with the number of clusters equal to 4. For sequence identifier I have used a two layered fully connected network with 100 neuons on each layer. I have experimented with different optimisers, losses and learning rates, but sadly enough, was not able to get satisfactory results. So i would not include a method in final comparison

3 Model Description

The techniques and experiments in related papers got me thinking about combining it with reinforcement learning techniques. Because reinforcement learning, is a type of machine learning in which an agent learns how to take actions in an environment in order to maximize a reward.

The environment consist of a share price graph with observing space size equal to number of embedding features (sets automatically on initialisation) plus current market values.

For the agent action space I chose 3 discrete numbers, representing an actions to buy, to sell or to stay out of market. As the agent observes the current state of the environment and chooses an action, the environment transitions to a new state, and also returns a reward that indicates the consequences of the action.

Environment also stores agent last prediction to measure the distance between real market data. The reward policy takes the distance and rewards if it does not exceed certain threshold. The environment is considered terminated when the relative position exceeds a threshold as well. In another words, when agent loses money.

The main idea behind Q-learning is that we maximize reward based on policy

$$\pi^*(s) = \text{argmax}_a Q^*(s,a), \forall s, \tag{1}$$

where π^* is our policy model.

Calculating the prediction loss in a DQN is a calculation of Temporal Difference error δ which derives from Bellman equation

$$\delta = Q(s,a) - [r + \gamma \max_a Q(s^{'}, a)]$$

Where the discount γ ensures the sum converges

$$\gamma \in [0,1]$$

We optimise prediction loss

$$L = \frac{1}{|B|} \sum_{s,a,s',r \in B} L(\delta), \tag{2}$$

via optimising Huber loss $L(\delta)$
Where B is a batch of transitions sampled from the replay memory

$$where \quad L(\delta) = \begin{cases} \frac{1}{2} * \delta^2, & \text{for } |\delta| <= 1, \\ |\delta| - \frac{1}{2}, & \text{otherwise.} \end{cases}$$

So, the value network is a neural network that takes input from the environment and returns a State-value function that evaluates how well the agent is in a given state. It evaluates the long-term value of a state. A policy network is a neural network that takes input from the environment and returns a probability distribution for discrete action that specifies the agent's strategy. The policy network allows the agent to choose the best action based on the received information about the state of the environment. During training, the policy network is used to select actions in the environment, while the target network is used to calculate the target values for the Q-learning update step. In this way, the target network provides a more stable estimate of the Q-values, which helps to prevent oscillation and divergence during training.

For the policy net architecture I firstly experimented with a FCN (fully connected neural network), and then applied LSTM (long short-term memory) approach with sequential input. And got interresting results.

4 Dataset

For the task of predicting stock market I needed a dataset with news articles and current market state. I have researched for avalable datasets, and there were plenty of twitter's related ones. I wanted to experiment on Russian stock market via using Russian news. And specifically, I did not chose the existing news datasets because they were too broad on topics. My goal was to process only financial related articles. So my choice was the Finam's newsline[1].

I should mention that, it was not easy to get it downloaded. My first naive approach was to just load it using python requests. But it turned out that they had an interface, where you need to click on the button in order to load another portion of news. So i discovered python selenium, a new approach to emulate human-like gestures and button clicks on the web page. The notebook `/notebooks/news-scraping.ipynb` consist of scraping a web page with news.

It is important to mention that the dataset is available for the research purposes. Since Finam allows to use the information with giving a link to it according to copyright policy Each row contains HTTP link to the source.

[1]Raw content from Finam's newline could be found here.

The second stage in the notebook `/notebooks/stocks-dataset.ipynb` was to load a stock market values from stock exchange. As an employee of Tinkoff bank and an active contributor of invest-python library I used the Tinkoff Investments OpenAPI [2] to load the Sberbank privileged shares with FIGI number `BBG0047315Y7` for the period of previous year. I chose the Sberbank shares instrument because it has high liquidity and high volatility. I believe that such instrument is quite popular among investors.

Having two separate datasets, I have joined them together by matching a `date` column that contained current date. See the details in `notebooks/dataset.ipynb`. The example can be seen in Fig. 1

	date	title	text	link	author	label	open	high	low	close	volume
0	2023-05-06	Инвесторы все сильнее верят в дефолт США через...	Джо Байден не хочет повышать потолок госдолга	https://www.finam.ru/publications/item/investo...	Митрофанов Никита	negative	235.83	240.93	233.42	237.06	11352.0
1	2023-05-06	Встреча по зерновой сделке на уровне заместите...	Встреча пройдет с участием России, Украины, Ту...	https://www.finam.ru/publications/item/vstrech...	Finam.ru	positive	235.83	240.93	233.42	237.06	11352.0
2	2023-05-06	Закрытие дивидендного реестра по "Сбербанку" с...	Последний день с рекордными дивидендами акции ...	https://www.finam.ru/publications/item/zakryti...	Кожухова Елена, ИК "Велес Капитал"	positive	235.83	240.93	233.42	237.06	11352.0

Figure 1: Dataset samples represented as DataFrame.

I represent some statistics of resulting StocksNews dataset on Tab. 1 The dataset was split into two parts: train and test with proportion of 88/22 percent accordingy. It's important to mention that this is a time series dataset so I did not use any kind of shuffling.

	Train	Test
Proportion	0.88	0.22
Size	23879	6735
Vocabulary size	16507	
Date range	2022.05.08 - 2023.02.10	2023.02.10 - 2023.05.06

Table 1: Statistics of the StocksNews dataset.

When constructing a baseline with TF-IDF[3] embeddings, I have used lowercase sentences, `mystem` for lemmatizaton, `nltk` for stopwords filtering.

It's worth mentioning that use of other dataset preprocessing techniques have not been done. It could be a topic for other possible research which is out of scope of current.

For implementing a related work solutions I have labeled every sentence with one of four sentiment labels which was correct in my personal opinion as a novice investor. The distibution of labels is shown of Pic. 3

[2]Tinkoff Invest OpenAPI could be found here.

[3]TF-IDF model to get the sentence embeddings. Details at Wikipedia or in work by [Ramos, 1999].

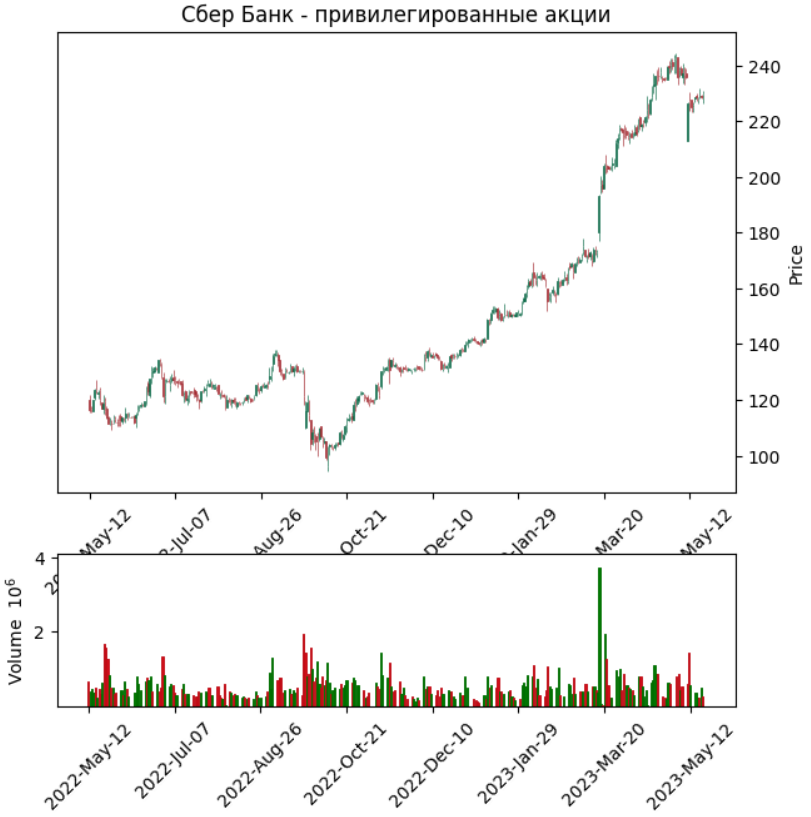

Figure 2: Sberbank shares cost values in dataset.

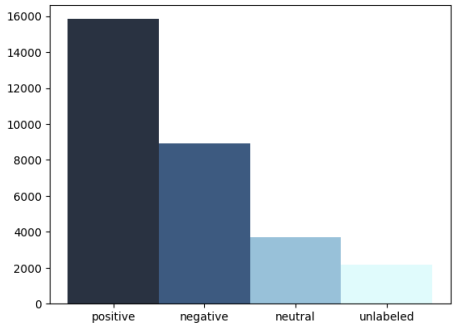

Figure 3: Distribution of sentiment labels in dataset.

5 Experiments

5.1 Metrics

The evaluation of every solution was scored by 2 types of metrics: regression and classification metrics. My goal was not only to see how close the forecast is to real data via checking regression metrics score, but also to check whether the predictions are correct in the terms of market change direction. And see if you can rely on it during trading. For that I have used classification metrics.

Following sections contains $\hat{y}$ and y as a forecast values and actual values respectively.

Regression metrics The first is Mean Squared Error:

$$MSE(y, \hat{y}) = \sum_{i=1}^{D} (y_i - \hat{y}_i)^2, \tag{3}$$

this metric was chosen to have a general sense of how close the forecasts are.

For intuitive interpretation the Mean Absolute Percentage Error was used:

$$MAPE(y, \hat{y}) = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{4}$$

Classification metrics The accuracy was also used to have a comprehension of models correct choices. I named it as Stock Option Accuracy Score:

$$SOAS(y, \hat{y}) = ACC(z, \hat{z}), \tag{5}$$

$$where \quad ACC(z, \hat{z}) = \frac{|\{z_i == \hat{z}_i \mid i = 0...n\}|}{n},$$

$$where \quad z_i = \begin{cases} 1, & \text{if } y_{i+1} - y_i > 0, \\ 0, & \text{otherwise.} \end{cases}$$

$$where \quad \hat{z}_i = \begin{cases} 1, & \text{if } \hat{y}_{i+1} - \hat{y}_i > 0, \\ 0, & \text{otherwise.} \end{cases}$$

I also came up with a metric called Stock Possible Profit Percent Score or S3PS for short:

$$S3PS(y, \hat{y}) = 100 * [\prod_{i=0}^{n-1} (1 + C_i) - 1], \tag{6}$$

$$where \quad C_i = \begin{cases} P_i, & \text{if } y_{i+1} - y_i > 0 \quad and \quad \hat{y}_{i+1} - \hat{y}_i > 0, \\ P_i, & \text{if } y_{i+1} - y_i <= 0 \quad and \quad \hat{y}_{i+1} - \hat{y}_i <= 0, \\ -P_i, & \text{otherwise.} \end{cases}$$

$$where \quad P_i = |1 - (y_{i+1}/y_i)|$$

It is based on SOAS but represents possible percentage gain if we follow a strategy suggested by model.

5.2 Experiment Setup

I setup the experiment and train the proposed model for 5000 episodes (epochs). The hyperparameters are show in Tab. 2 It was convinient to run training procedure in jupyter notebook, so I represented learning algorithm on the graph as shown in `/notebooks/lstm-gym-embeddings.ipynb`

I also experimented with bert embeddings using sentence transformers and rubert-tiny pretraind model.

Hyperparameter List		
Param Name	value	description
$BATCH - SIZE$	128	bathes from the replay buffer
$FAIL - DIFF$	0.05	max agent deviation to terminate env
$DIFF - REWARD$	0.001	how close agent should be to the target to get a reward
$SEQUENCE - LEN$	42	length of data for previous days
$GAMMA$	0.99	discount factor
$EPS - START$	0.9	start of epsilon
$EPS - END$	0.05	final value of epsilon
$EPS - DECAY$	1000	rate of exponential decay
TAU	0.005	update rate of the target network
LR	1e-4	learning rate

Table 2: Hyperparameters of proposed method.

5.3 Baselines

First of all I implemented a baselines with TF-IDF embeddings and multiple linear or non-linear methods over it. I got some results to start with and some functions which helped me in the future solutions. For more info, please visit a `/notebooks/baseline.ipynb`

6 Results

In the Tab. 3 I refer to achieved results and provide brief conclusions.

Results comparison				
Method Name	MSE	MAPE	SOAS	S3PS
RIDGE	1253.37	0.12	0.44	4.85%
TREES	3919.90	0.25	0.46	-14.12%
LSTM	155.04	0.04	0.41	4.24%
DQNFCN	2027.74	0.20	0.44	-9.75%
DQNLSTM	166.67	0.05	0.56	7.33%

Table 3: Results of all methods of research.

In the course of experimenting I implemented a prototype for each solution and graphical representation for better understanding of the results.

A baselines are good starting points, because some of them have reached the FCN proposed solution.

Even though the proposed model did not overcome the LSTM by [Sen and Mehtab, 2021] in regression metrics, it showed a decent results in classification metrics. S3PS metric shows how much percent we could possibly gain when following a strategy. With that being said, it means the reinforcement learning techniques can be useful for solving a human-like tasks, including trading.

The sample of model forecast results could be found in Fig. 4.

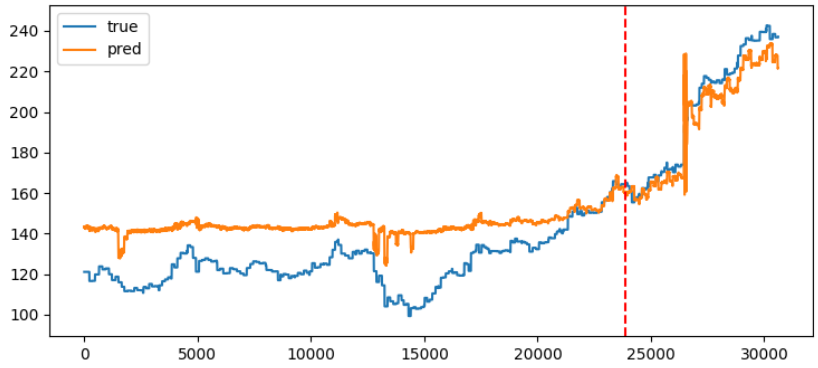

Figure 4: Price forecast split by red dashed line.

7 Conclusion

It was quite interesting research for me, I have managed to make my own dataset4, solved the task with baselines5.3, implement available solutions using research reports which were mentioned in section2, explore gymnasium deep q-learning framework, and make my own RL environment where I did all the experiments5.

It's worth mentioning that there are plenty of other research possibilities left out of scope of current work, including hyperparameter tuning, model ensembles, different architectures, also sources of news articles, text preprocessing, other companies shares and so on.

To conclude, reinforcement learning is a powerful machine learning technique that has shown great promise in solving complex real-world problems.

References

[Bollen et al., 2011] Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

[Hartigan and Wong, 1979] Hartigan, J. A. and Wong, M. A. (1979). A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Ramos, 1999] Ramos, J. (1999). Using tf-idf to determine word relevance in document queries.

[Salimi-Badr and Ebadzadeh, 2022] Salimi-Badr, A. and Ebadzadeh, M. M. (2022). A novel self-organizing fuzzy neural network to learn and mimic habitual sequential tasks. *IEEE Transactions on Cybernetics*, 52(1):323–332.

[Sen and Mehtab, 2021] Sen, J. and Mehtab, S. (2021). A robust predictive model for stock price prediction using deep learning and natural language processing.