SI630 Project Final Report

Chongdan Pan
pandapcd@umich.edu

Xinyi Ye
sylva@umich.edu

Abstract

This report is for a SI630 (Natural Language Processing) Project. It covers the main idea, technique as well as models used in constructing a profitable investment strategy in cryptocurrencies market behavior based on text gathered from Twitter. Then experiments are designed to evaluate the performance of the models as well as help generate insights about NLP's role in the finance area.

1 Introduction

For now, Natural Language Processing has become the hottest topic in the technology area, because it can be used as a great tool to understand human behavior. For example, NLP can help us understand the emotion of individuals and generate some insights into collective behavior.

The typical collective behavior of humans is trading in a market. People have used different methods such as graphs and quantitative models to understand the market's behavior, and we believe NLP can definitely be a useful tool. A lot of financial institutions have set up teams trying to dig some valuable information from the market through NLP. For traders and investors, NLP may be used for them to find out the hot spots in the market and make a profit. For governors, NLP can be used to understand the emotion of the market and evaluate the risk.

In traditional finance area, people typically use time-series model such as ARMA or Garch models to fit the return of certain asset. It's not until recently that experts turn to their glare to machine learning model. NLP, as a hot topic in machine learning, aiming at capture the information of market by text. The text can contain market's emotion, or some news that may have a significant impact on the return. Therefore, this project will do some exploratory analysis based on NLP technology, and try to dig some valuable information out of the market. The task is quite valuable, especially for people who are interested in investment. As far as we know, there are a lot of institutions are trying use NLP model to gather market information to generate higher excess return. The task is also very helpful for people to analyze the market and understand people's interaction with it.

The input of the task can be any text related to the market, such as comments, articles. In our approach, we use a Word2Vec model to generate a distributional representation for the words, and use LSTM and RNN to model the return of the asset. It's worth mentioning that all these texts should appear before the market's behavior, otherwise they're just the feedback of the market behaviors. The output will be the predicted return of the asset, and we'll make investment decision based on the predicted return. It turns out that LSTM works better than RNN as well as traditional ARMA model, thanks to its ability of memorizing long-term information.

In this project, we apply a series of NLP models to a real problem that worth efforts. We start with data gathering and cleaning, and then use different models to represent the text information, lastly we successfully to generate profitable investment strategy. In technique area, we get experience in building a machine learning pipeline for a real problem, while in application, we may actually identify some way to make profit by the model.

2 Data

We have data from two Twitter API and Binance market data API. The time range of the data is from 2021/01/01 to 2022/04/01.

2.1 Twitter Text

We're using the text data from Twitter to train our model. Since there are too much text on twitter everyday, and we our computer can process all of them, we use a filter when querying the data from

Twitter API. Since Twitter have a limitation on how much data you can get every month, our query must be very specific. We only query for data with a hash tag ETH or Ethereum, and there is number and dollar symbol in the tweets. Besides the text, we also get the timestamp of each tweet so that we can match it with the market data.

Besides the text information, tweets may have other entities such as emoji or entities, therefore, we're using regular expression to extract the information that we're interested in. After counting all the words in the corpus, we find the most frequent words are coin's name or some stopwords, which is meaningless. On the other hand, we also find that the lest frequent words are some meaningless components of URL. Therefore, we're going to set some threshold on the frequency of words and drop out those don't have much meaning.

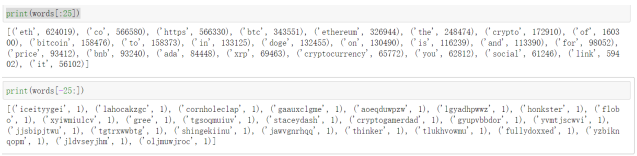


Figure 1: Most frequent and lest frequent words

2.2 Binance Market Data

For market data for ETH, we're using Binance's API and fetch candlesticks of every 1 hour. For simplicity, we're only considering the close price of each interval, and we also keep other fields such as trade volume, buy volume as well as timestamp.

To make profit, we focus more on the return and other property of an asset rather than the price, therefore, we calculate the return manually by $R_t = \frac{P_{t+1}}{P_t} - 1$, and our goal is trying to find some causality between the text and return.

We've plot the histogram of the return, and it has a range from -0.13 to 0.07 with a mean to be -0.001. Due to the heavy tail of the return, it follows a t-distribution

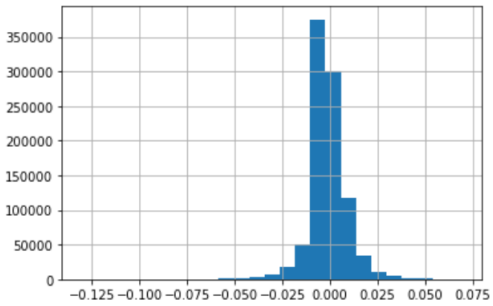
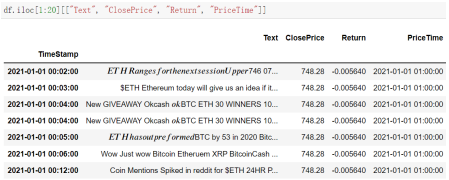


Figure 2: Distribution of return

2.3 Sample Training Data

For now, our goal is to use all the text related to ETH price posted on Twitter to predict the return of next hour. Therefore, we're merging the data by timestamp so that the text always happen before the return.


df.iloc[1:20][["Text", "ClosePrice", "Return", "PriceTime"]]

TimeStamp	Text	ClosePrice	Return	PriceTime
2021-01-01 00:02:00	*ETH Ranges for the next session Upper* 746 07...	748.28	-0.005640	2021-01-01 01:00:00
2021-01-01 00:03:00	$ETH Ethereum today will give us an idea if it...	748.28	-0.005640	2021-01-01 01:00:00
2021-01-01 00:04:00	New GIVEAWAY Okcash ok/BTC ETH 30 WINNERS 10...	748.28	-0.005640	2021-01-01 01:00:00
2021-01-01 00:04:00	New GIVEAWAY Okcash ok/BTC ETH 30 WINNERS 10...	748.28	-0.005640	2021-01-01 01:00:00
2021-01-01 00:05:00	*ETH has outperformed* BTC by 53 in 2020 Bitc...	748.28	-0.005640	2021-01-01 01:00:00
2021-01-01 00:06:00	Wow Just wow Bitcoin Ethereum XRP BitcoinCash ...	748.28	-0.005640	2021-01-01 01:00:00
2021-01-01 00:12:00	Coin Mentions Spiked in reddit for $ETH 24HR P...	748.28	-0.005640	2021-01-01 01:00:00

Figure 3: Sample Data

As shown in the figure, the data from training can be text posted from the first hour in 2021/01/01, but we're going to predict the return in the second second hour. In our project, we're going to use all data from 2021 for training and validate our model with data from 2022.

3 Related Work

1. (Shahzad et al., 2021). This paper used Linear Regression, Deep Neuron Network as well as LSTM to predict the price of bitcoin-based on tweets. It uses sentiment analysis to see whether the tweets have a positive or negative effect on bitcoin's price. However, it doesn't get me any clear result or conclusion about each model's performance. On the other hand, it only considers the plain text of each tweet, while I think we also need to consider who posts the tweets. Therefore, I may work more on the preprocessing and evaluate the performance of various models.

2. (Laskowski and Kim, 2016). This paper was published in 2016, where bitcoin and blockchain are not quite popular right now. It focuses on getting data from social media

and looking for the correlation between bitcoin price and specific hashtags. It turns out bitcoin-price talk has a much higher correlation to bitcoin than other channels. This result can give me some insight into how to choose keywords for getting the data. On the other hand, it also gives me a rough idea about the size of the data as well as the time and memory required for processing. In this paper, the data size is 200GB and uses 1.95 GB RAM on average. However, I think it's a reasonable size for my project since I won't get data from so many channels and computers are more advanced now.

3. (Huang et al., 2021). This paper focuses on applying LSTM on sentiment analysis of people's posts on Weibo, one of the biggest social media in China. Similar to other papers, LSTM is used to predict if the price will go up or go down. The most interesting part of this paper is that they construct a vocabulary set with unique characteristics related to the crypto market. However, they manually construct the vocabulary set and identify the keywords related to the crypto market, which looks quite inefficient to me. Therefore, I think TF-IDF probably can be useful for us to identify the key pattern.

4. (Wong, 2021). This paper used Naive Bayes and LSTM model to do sentiment analysis. It turns out Naive Bayes does a better job than LSTM in distinguishing similar tweets' relation to the crypto prices. However, there is no general threshold for these two models to do a good classification on whether the tweet is negative or positive, which means these models are better at doing a relative comparison between two tweets. As for classification, LSTM has 51% accuracy while Naive Bayes only has 50%, which means they're not working well. The result looks much more like a random guess, but it can serve as a baseline for the project.

5. (Patel et al., 2020) This paper used a LSTM and GRU-based hybrid cryptocurrency prediction scheme to predict two cryptocurrencies, namely Litecoin and Monero. LSTM has been proved to be the best in until now due to their ability to remember and extract the temporal features of data. The results depict that the proposed scheme accurately predicts the prices with high accuracy, revealing that the scheme can be applicable in various cryptocurrencies price predictions.

4 Methods

1. Lemma Extraction
   There are a lot of noisy text in the tweets due to the natural of this social media. After applying regular expression on these text, we think there are still meaningless words in our data, such as stopwords. On the other hand, we believe the key information are contained within the lemma of the text. For example, "buying" and "buy" won't make a difference in the information. Therefore, we used Spacy to remove the tradition stopwords, and only keep the lemma of remaining words. Words like "buy", "buying", "bought" will all be treated as "buy". Such preprocessing can help us decrease the size of our corpus and accelerate the training process for the models.

2. Embedding the words with word vectors
   Word embedding is one of the most popular representation of document vocabulary, and we're using the basic Skip Gram algorithm in our project. The main idea is to use vector of certain length to represent each words. In essence, we're training a neuron network to predict the probability of appearance of each word and its context words. The vector representation for each word will be initialize by one hot encoding, which has one on the element have the same index with the word and zero else where. Then we input the word of interest and its context words, the model's prediction should be one since they appear in the same context. After being trained for one epoch, the model has a high accuracy, and its parameter will be the vector representation of each word. Based on the idea of Word2Vec, the dot product of the vector of words appearing in similar context should be close to 1 hence they're close in meaning. We can also use this way to find out the how similar words are.

   Besides the basic training process, we also use subsampling, negative sampling, better UNKing to improve the training procedure. In subsampling, we're delete some frequent

words in the corpus so that each word has fair appearance in the training data. For negative sampling, we're using words out of the context to generate negative sample, which means the model's output for probability should be 0. Such method can generate samples with different labels and can accelerate the training. Better UNKing treats different unknown tokens differently. Although our model may encounter unknown words, it still can extract some information based on the prefix or suffix of the words, therefore, we want to encode the unknown words differently to capture the information.

3. Use sequence model such RNN and LSTM
   A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. Sequential data is basically just ordered data in which related things follow each other. In a RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously. That is to say, Every prediction at time t (h_t) is dependent on all previous predictions and the information learned from them.

   Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory. The LSTM consists of three parts. The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

   Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by C(t-1) and C(t) for previous and current timestamp respectively. The cell state carries the information along with all the timestamps.

   In our method, we'll build both RNN and LSTM which take the vector representation of each words and generate the predicted return of each timestamp. We'll sort all tweets based on timestamp, and feed them to our model together. At the timestamp when we want to make a prediction, we'll check the model and see its prediction. Note that our model doesn't only memorize the input of current sentence, but all previous sentences, because the information from one month ago may still affect current return.

5 Evaluation and Results

We're using two models as baseline, which are traditional ARMA model which only taking the previous return as input and a linear regression model, which take the average of all word vectors in the prediction period as input. For evaluation, we're using two metrics, the first is Mean Square Error for the return, which is (Final value)/(Initial value)-1 the second is how much profit we get based on our model's prediction. For the second metrics, we're actually simulate a investment process based on the output of our model. Assume we have one token at the beginning, if the model has a negative prediction for return, we'll sell it into cash, and we'll spend all our cash buy one if the prediction is positive. Therefore, we can keep selling and buying based on the models' prediction. In the end, we'll compare how much value we have to the market value of the token. The excess return is (Asset value in the end)/(Market value in the end) - 1.

In addition, we'll run our model on different cryptocurrency and make decisions on different time interval. We also gathered data for a less popular cryptocurrency AAVE for training. The difference in the models' performance on different cryptocurrencies can help us understand the generality and robustness of our model. We'll ask our model to make prediction every one hour and four hours as well to see if the trading frequency will affect the behavior of our model. With higher frequency, we have less information for the model to make decision, but it also have more opportunities to react to the market.

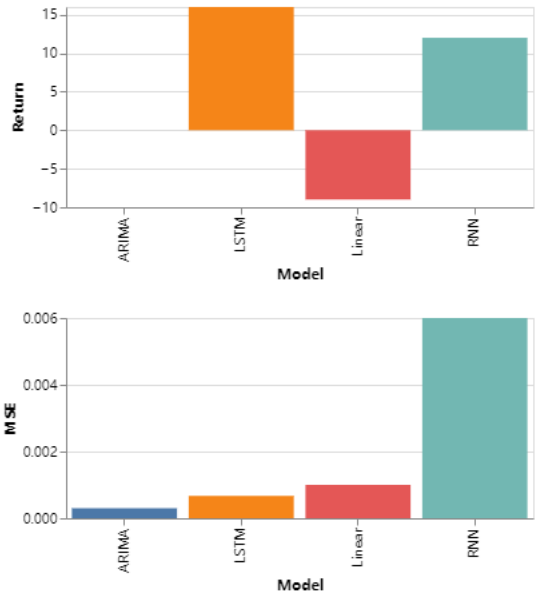


Figure 4: Metrics on different models

Model	Token	Freq	MSE	Ret
ARMA(4,4)	ETH	4H	3E-4	0%
Linear	ETH	4H	1E-3	-9%
RNN	ETH	4H	3E-3	12%
LSTM	ETH	4H	6.7E-4	16%
LSTM	ETH	1H	2.9E-4	19%
LSTM	AAVE	4H	1.4E-3	23%
LSTM	AAVE	1H	1.2E-3	25%

First, we're running all models on the data of Ethereum and make decision based on prediction every four hours. From the plot and table we can see each model's characteristics. Our baseline models, ARMA(4,4) and Linear Regression can't bring us any profit. The ARMA model can't give us any return because it's prediction is always 0 so that we don't take any action. This is probably due to the ARMA model is stationary, and it has already been in the stable phase. Although ARMA has lowest MSE, it won't help us make investment a profitable decision, implying the importance of using return as our metrics. The linear model even lead to a negative return, we believe it's because the model is just doing some random guess, so it's hard for it to capture useful information.

Figure 5: LSTM performance on ETH 4-hour data

Both RNN and LSTM can give us positive return, but LSTM has lower MSE as well as higher return. It makes perfect sense because LSTM can do anything that RNN can do, and LSTM can memorize more useful information. From the figure 5, we can see that the asset value based on our model can have a significant advantage over the price of the token itself. However, we're also interested in change the hyperparameters such as the frequency to make a decision or see how the model can be generalized to other tokens.

Figure 6: LSTM performance on AAVE 4-hour data

Hence, we apply the LSTM model on another token AAVE's data, and it turns out that we have much higher return. From the plot, it looks like that our model successfully escaping from a large drop in Jan. 23rd. We think it's because AAVE has much less data so our model won't give a positive prediction at that moment so easily. Then we stick to ETH and decrease the interval from 4 hours to be 1 hour, which means that the model have less information to make decision more frequently.

Figure 7: LSTM performance on ETH 1-hour data

From figure 7, it turns out that LSTM model will be even more profitable when working on high

frequency data. Since we already have extremely large amount of text data, decreasing the decision-making interval won't reduce the amount of information. On the other hand, since the model is required to make more decision, it has more chances to show its profitability as well.

6 Discussion

Our approach did well overall. The performance is generally satisfying for an end-user of our NLP model. The baseline model is linear regression. The MSE is 1e-3 and return is -9%. We tried three other models, which are ARMA, RNN and LSTM. Their respective MSE are 3e-4, 3e-3 and 6.7e-4. Although some models don't outperform the based model from the MSE respective, they all achieve higher return. Among them, LSTM performs the best. The final return rate is positive, which means that we can actually earn money using this strategy. The performance is close to our expectation. The improvement of is due to the fact that the prices of the cryptocurrencies have obvious time series patterns. The prices are highly related to the news and information several days ago, or even several months ago. LSTM aims at solving the problem of gradient disappearance and gradient explosion in long sequence training, which is quite suitable for our specific problems. That's why it performs best here. RNN performs a little worse than LSTM, it is because RNN can only have shortterm memory, and it does not take longterm memory into consideration. In conclusion, there are three reasons that our strategy is good. First, the data we gathered is indeed has contribution to the prices of the cryptocurrencies. Second, the way that we embedded the data can actually catch and represent most of the information in the datasets.Third, our model takes full use of the patterns of the data.

7 Conclusion

For the project, our goal is to use some NLP models to construct a profitable investment strategy for investing cryptocurrencies. Our method includes four stages. First we used regular expression and pos tagging to extract the key text of the tweets. Second, we trained a Word2Vec model to generate distributional representation. Third, we split the data. The data in 2021 will be used for training, and 2022 will be used for prediction. At last, we trained different models, i.e. linear regression, ARMA model, RNN model and LSTM model and looked into their performance. From the results we can find that LSTM has the best performance. We can actually earn money from the strategy we constructed.

8 Other Things We Tried

We tried data with higher frequencies, i.e. gathering data for every second. However, the amount of data is so large that the computer does not work when filtering and processing the data. We tried to run it on Hadoop clusters. We first built the cluster. However, we met with a lot of problems when building the cluster, such as the dismatch of Java version and the contents of the configuration files. We worked on it for one week, but it is still not working. We also tried Transformer model. We used an open-source version on GitHub. We spent nearly a week to configure the environment and we spent another week to learn its mechanism and to figure out how to apply the model to our specific problems. It took much time to train and tune the parameters to get the good results. Finally, we did not have times to do it.

9 What You Would Have Done Different or Next

For the data part, if we started the project over, we will crawl the data not only using the crypto tags, but will gather data with wider variety. For example, we will gather the data in different types of forums corresponding with cryptos, as well as data in different news websites. We will also embed market data and text together as input, such as data of other cryptos. For the model part, we can improve in two aspects. First, we can add the influence of the spokesmen. For example, we can use a hyperparameter to represent the weight of each text, and train it together with other hyperparameters. Second, we can use some more advanced models, such as BERT and GPT. The above two ideas are the ones that we have wanted to try but didn't have time. Our results show that the returns are not only related to the news from the day before, but are also related to the news from several days ago. That's why LSTM performs better than RNN. Therefore, we can introduce the attention layer to resolve the problem of long distance messages being weakened.

10 Group Effort

The division of the work is
Chongdan Pan:

1. Getting the data. He first got familiar with twitter's API and got historical tweets, then used them to crawl the data.

2. Language Processing. We applied different methods to process the data and turned it into some matrix representation that can be directly sent to the neuron network model. He built the termdocument matrix to do the linear regression and logistic regression. He also built the Word2Vec to encode the data.

3. Model Design and Training. He built the linear regression and logistic regression. To make the model better, he built the LSTM model and changed some optimizers and training hyperparameters.

Xinyi Ye:

1. Language Processing. We applied different methods to process the data and turned it into some matrix representation that can be directly sent to the neuron network model. She built the Word2Vec model to encode the data.

2. Model Design and Training. She built the ARMA model and tuned the hyperparameters. She built the RNN model and changed some optimizers and training hyperparameters.

References

Xin Huang, Wenbin Zhang, Xuejiao Tang, Mingli Zhang, Jayachander Surbiryala, Vasileios Iosifidis, Zhen Liu, and Ji Zhang. 2021. Lstm based sentiment analysis for cryptocurrency prediction.

Marek Laskowski and Henry M. Kim. 2016. Rapid prototyping of a text mining application for cryptocurrency market intelligence. In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, pages 448–453.

Mohil Maheshkumar Patel, Sudeep Tanwar, Rajesh Gupta, and Neeraj Kumar. 2020. A deep learning-based cryptocurrency price prediction scheme for financial institutions. *Journal of information security and applications*, 55:102583.

Muhammad K. Shahzad, Laiba Bukhari, Tayyeba Muhammad Khan, S. M. Riazul Islam, Mahmud Hossain, and Kyung-Sup Kwak. 2021. Bpte: Bitcoin price prediction and trend examination using twitter sentiment analysis. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 119–122.

Eugene Lu Xian Wong. 2021. Prediction of bitcoin prices using twitter data and natural language processing.