Data Scraping from Twitter and Cryptocurrency Exchanges, Sentiment Analysis and Prediction for Cryptocurrency Trading and Arbitrage

Anant Bhargarava, Prakhar Kaushik
Johns Hopkins University
EN.601.466 Final Project

May 19, 2018

Abstract

Cryptocurrency markets have been the harbingers of great turmoil to international financial markets by bringing a highly volatile and decentralized notion of a financial instrument. With the highly distributed nature of crypto-currency trading, and the ever increasing number of currencies and exchanges, it has opened up new avenues for investment and profit. A predictive algorithm which could predict and inform us when and what to invest depending on the financial options we have at that point of time would go a long way in ensuring profits for crypto-currency traders.

1 Background

Our project consists of two major parts - data extraction and data analysis. In the first part, we scraped cryptocurrency data - exchanges, coins, price, historical price, traded volume etc. and relevant tweet data for processing. We, then, processed these both raw data sets and produced information which is then processed and combined. The user inputs the currencies that our network then uses this data and uses sentiment analysis to predicts whether our trades will be profitable in the coming future (30 minutes for the data we used). Though, we present only one instance and one pair (USD-BTC) as a working example for our project, our concept and program can be extended and used for multiple currencies at once as well as longer future predictions.

1.1 Cryptocurrency Data Extraction

For purposes essential to our project, we required access to cryptocurrency data from multiple exchanges. The data included types of coins, exchanges, currency-pair values, realtime currency prices, trading pairs, historical volume and price data, etc. We can collect all this data by using APIs from each of the exchanges accessible to us and accessing their data individually. We can also use certain APIs or libraries which combine the APIs from all these different exchanges and provide us a single interface for extracting the data. Initially, we used the *ccxt* library for our purposes - this worked efficiently, but required us to create accounts in all of the exchanges and acquire API keys for each of them. We did that, but in order to present our work more visually, we used an API - *Cryptocompare*, which allows us to access all the required data from all the exchanges without using API keys for each of them. We have created a program which allows the user to enter the currency or currencies he has as well as the currencies he/she wants to trade in. For example, the user may have USD, BTC and INR at a point

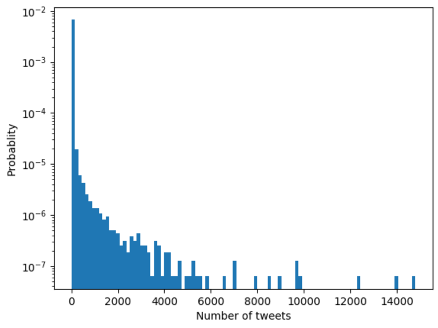

Figure 1: Number of tweets by twitter handle

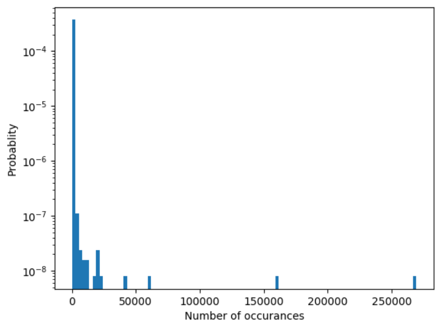

Figure 2: Occurrences of same hash tags

in time and may want to trade only in ETH, BTC and Ripple. The user can also view the number of exchanges available, the number of exchanges available for his trade pairs, coins available, regular currency trade pairs, currency trade volume (present and historical) and so forth. This allows the user the leeway to choose whatever exchange and coin he likes and wants and has the data to make an educated guess (for e.g. the program also has a functionality for providing us a list of exchanges sorted by trade volume for the trade-pairs in question). Our code also provides a matrix of prices and volumes for our currency (our trade pair) prices and trade volume time-indexed (hourly in our experiment. For our experiment, we used a single pair USD-BTC and passed hourly price-volume values to our deep network. We note, that we are only working on a single pair, our concept and code can be used for n number of currency trade pairs.

1.2 Preprocessing For Neural Network

For purposes of training tweets regarding bitcoin were used from the [https://github.com/meetyildiz/tweetwise]. It had 1mn tweets distributed in 2800 time segments spread with 1hr time difference between different time points. For each time segment, tweets at least half hour from the final time were selected.

Tweet contains the identity of the handle which tweeted along with the entire text and time stamp of the tweet. Very few hashtags are popular and very few handles tweet frequently about BitCoin. So 200 most frequent twitter handles as well as most frequent hash tags were selected from the data. For each tweet it was noted if it contains the selected hash tags, and one hot encoding was done for the handle if it was one of the frequent handles.

Along with tweet data, data for the past 200 hours is also used.To ensure that the computation can be done in batch, number of tweets used per time step is limited to 300 per time step, and number of words per tweet is limited to 30. If the number of tweets in the time step is less than 300, or the number of words in the tweet is less than 30 then extra padding with zeros is done. The padding should have no effect on the network, as max for each output layer of the convolution is taken. This also, makes the network intolerant to number of words or number of tweets.

1.3 Neural Network Architecture

The architecture is composed of 1D convolutional layers with different kernel sizes to capture relationships across different number of tweets. In the current variant 30 kernels are used with sizes 1,2,3,4,5,6 and 8. Global max pooling is done across each output layer of the convolution. The outputs are concatenated in a single layer, along with output from a linear layer

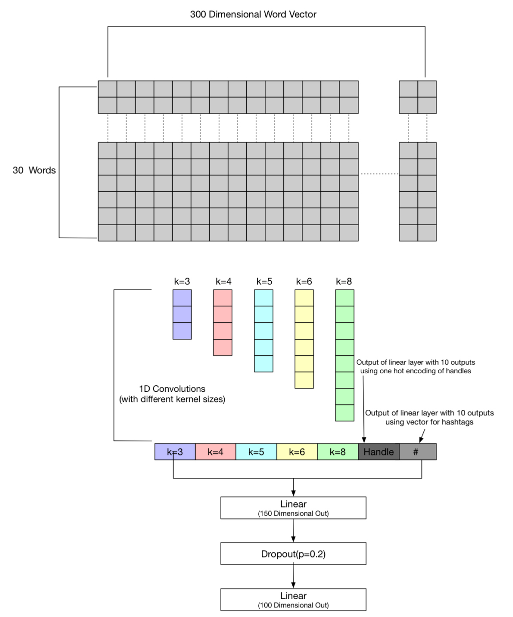
300 Dimensional Word Vector
30 Words
k=3
k=4
k=5
k=6
k=8
1D Convolutions
(with different kernel sizes)
Output of linear layer with 10 outputs
using one hot encoding of handles
Output of linear layer with 10 outputs
using vector for hashtags
k=3
k=4
k=5
k=6
k=8
Handle
#
Linear
(150 Dimensional Out)
Dropout(p=0.2)
Linear
(100 Dimensional Out)

Figure 3: Architecture for analyzing sentiment for each tweet

for handles and hashtags. Two linear layers are used after the concatenation to make a 100 length vector representing each tweet.

Outputs from sentiment analysis of all tweets is concatenated. Convolutional layers with varying kernel sizes of 4, 8, 16, 32, 64, and 128 are used to extract features from the analysis and global maximum across each output channel is taken. LSTM with 100 unit output is used to get historical trend based on the volume and price of bitcoin for 500 hours prior to the current event.

For all linear layers ReLU activation is used. Binary cross-entropy is used as loss function. Adam Optimizer is used for training with learning rate of 0.001. Regularization done using dropout and early stopping.

2 Results

In our code, we have used a specific case (USD-BTC) pair for our calculations and predictions. Our concept can be easily extended and used for other and multiple trading pairs. This example, though, is enough to represent the concept effectively. Our code for cryptocurrency data extraction is capable of extracting all data from about 120 cryptocurrency exchanges as well as about 200 cryptocurrencies. The program processes the data and gives the user an opportunity to access all of it in the way he/she wants. Our code has the capability to induct a lot more components than we have displayed (these can be added according to the users choice).

3 Conclusion

After, experimenting with a lot of concepts for a problem, we realised that the volatility of the cryptocurrency market is highly correlated the buyers' sentiment. A good measure of this sentiment can be analyzed through social networks. The only drawback being that it is not the only thing that the market depends upon. Our code can be thought of a framework for building upon crypto prediction, especially for arbitrage purposes. Adding more data and better and more effective filters will increase the prediction rate of the network. This program is one of the the first efforts to predict cryptocurrency rates and use them for trading.

Current accuracy we achieved is 67%. This can be improved by gathering more data and analyzing the twitter data much more along with doing network analysis on the twitter data.

Future work involves building dependence trees as well broadening of input data base. We can build weak dynamic weak learners with our different categories of inputs in order to manage dynamic data sources.

4 Appendix

All code should be run in python3 Libraries required for code to run: Keras, tensorflow, ujson, numpy, spacy, pandas, sns, ccxt,configparser,numpy, matplotlib, ujson, seaborn, tweepy

To pre process the data approximately 1hr on 8 core CPU would be required, this is due to calling spacy for each term to get the word vector. File $tweets_{r}aw.txt$ from the repository[$https://github.com/meetyildiz/tweetwise$], and $handle_{2}00.txt$, $hashtag_{2}00.txt$,$prices.txt$, should be placed in data folder before doing the preprocessing.

We have two programs for cryptocurrency data mining - *arbitprime and WorkModel*. We need to run **WorkModel** program with the following syntax (use *help* to understand the syntax):
$python3\ WorkModel.py - - currency\ USD - - historical\ 200 - - trade - currency\ BTC - - print - ex\ T - - top - ex\ T - - coins\ T$.
Note- Please access *Help* before running the program in order to understand the requirements.
The program can work without the additional command line requirements as well and will output a $2x500$ *numpy* array of price and volume for BTC. Our code is hourly indexed (taking 500 hours of historical data by default)