Nexo crypto task

Petar Ulev

January 30, 2022

1 Introduction

This is a short report about the crypto task given by Nexo as a challenge. Here, I will present my results, interpret them and show visualizations, show the code structure and talk about further tuning of the code and some other ideas.

1.1 What is the challenge

The challenge is to connect to the Reddit API (https://www.reddit.com/dev/api/) and extract all threads that are related to to cryptocurrencies and are of today (for example: https://www.reddit.com/r/CryptoCurrency). Then, filter threads that are related to 3 coins (for example, BTC, ETH, ADA). Furthermore, analyse the filtered threads using different statistics (and possibly NLTK), analyse the sentiment of the different coins. Also, indicate how to automate this process to show results on daily basis.

2 My approach

As communicated and agreed with you, I am using the 'praw' library which is used to establish the connection with reddit API instead of using the classical requests approach. With a correct configuration file, the connection is made and I extract the top daily subreddits and preprocess them so they are convenient for analysis. I only take those subreddits that are related to the three specified cryptocurrencies, the others are filtered out.
The way I do the analysis is the following: first, I filter out the subreddits that do not talk about any of the specified three cryptocurrencies. After that, for each subreddit that we are left with, I save the title and all the top level comments. I go through all of them and I search for the specified cryptocurrencies. The sentiment analysis is made only on those comments/titles that contain at least one of the specified coins. For example, if a title or a comment talks about bitcoin, and we are interested in [bitcoin, ethereuem, cardano], we will consider that comment. Even if the comment talks about other cryptos but at least one in the list that we are interested in, we will consider it for the sentiment analysis.

Now, brainstorming about the task, it's not hard to see that this is an unsupervised problem - we don't have any labels/information about the subreddits, but we have the bare texts that we should analyse. This means that we should either use a pre-trained sentiment model, or use some kind of unsupervised algorithm (for example k-means clustering) to cluster the subreddits into sentiments. But the latter is less likely to work without careful preprocessing and further analysis, which could take more time. So I have decided to use NLTK's Vader SentimentIntensityAnalyzer. As described in their github, "VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media"[1]. The software is particularly good for this task, because it considers unprocessed text as it is. This means that punctuation, capital letters, repetitive letters that usually users use to strenghten their point/opinion, will actually be considered by VADER analyser. So I have tried to do minimalistic preprocesing on the data when I pass it to VADER, but I have actually heavily preprocessed it using NLTK while doing other analysis. The preprocessing includes tokenization, lemmatization, stop words removal, removing words containing non-letter characters, and others, all preprocessing is done using NLTK.
Apart from the sentiment analysis, I have, as required, done other analysis on the full text of the comments and it will be seen below.

3 Results and visualizations

For this results section, I consider the following list of cryptocurrencies: bitcoin, ethereum and dogecoin (these can be customly specified in one of the configuration files). As you can see in the below graphs for the sentiment - these currencies seem to have rather positive sentiment, although there are some negative and neutral sentiments as well. A more sophisticated analysis could be done by normalizing and reconsidering these values, so that we consider only highly positive sentiments (for example, 80 percent positive sentiment is something we would consider as "good" and "secure" to buy, potentially). Please note that all figures are moved in the bottom of the report because of LaTeX specifications.

3.1 Word count

Word count of each sample (subreddit in our case) can be useful for some other analysis, so I decided to include in anyway (check Figure 1: Word count)
You can see that most of the subreddits and their comments are around 400 words.

3.2 Cryptocurrency count

This is also an interesting statistic. It shows how many times each cryptocurrency appears in all of the subreddits (check Figure 2: Coin occurences)

It can be seen which cryptocurrency is the most talked about one, in our case - it's the bitcoin (as expected).

3.3 All word frequencies

This is a rather interesting statistic. It shows the top frequency words across all subreddits that we look at. We can further brainstorm how these words relate to the coins. Look at Figure 3: All word frequencies
You can see that words as "moon", "buy" and "like" can play a role here.

3.4 Sentiment analysis

Let's now see some sentiment analysis on the cryptocurrencies. Please have a look at Figure 4: Sentiment analysis on all cryptocurrencies.
As you can see, there are mostly positive sentiments of these three coins. Let's see the independent statistics on each one of them using pie charts

3.5 Pie chart for bitcoin

You can see the percentages of positive, negative and neutral sentiments as a pie chart on figure 5: Pie chart for bitcoin.

3.6 Pie chart for ethereum

This is the same chart as the one above, but for ethereum: Figure 6: Pie chart for ethereum.

4 Code structure

There is a main.py file, which is the entry point of the program and calls utils.py where the most of the analysis functions reside. I have also made a main.ipynb notebook that you can run cell-wise and inspect each different cell, change parameters, and so on. I have created a README.md file which explains in detail how to run the code. There is also a visualization folder - this folder contains all the graphs that are generated after the code is run. I advise that all graphs are deleted before a new run (this can also be automated). Also, please note that when running main.ipynb, some of the graphs won't be generated since the main purpose of the notebook is to see the graphs on the run. So if you want all the graphs - it's best to run main.py. I also have a requirements.txt file, this is also described in the README.md.
There is also a data folder, which is about the configuration files, but this is also described in the README.md file.

5 Room for improvement

I wanted to create a separate section where I talk about what improvements on the current approach can be done.
First and foremost, I think in most of the cases - especially for sentiment analysis, a supervised approach would heavily outperform an unsupervised one. So my idea is: take some comments/titles (I thin at least 5000), and manually label them as positive/negative/neutral (the more training data we have, the better). After that, we can do the classical NLP task: preprocess the sequences, encode the input sequences/texts using BERT (for example), and pass the encodings to a neural network or any other classification algorithm to train. For new data - we do the same - preprocess, encode, input into the model.
Next, I think it can be useful to consider comments/titles that somehow relate to the cryptos, but don't contain the crypto word explicitly. For example, if there is a subreddit with positive title about bitcoin, and a comment "I don't this this is true", we should consider this as a negative sentiment, but currently we will just ignore the comment because it doesn't contain the word 'bitcoin'.
Another thing that I was considering is that maybe titles should have more importance than comments.

6 Automation question

There was also a question about automation - given the working script, how to automate it so that information is updated on daily basis. This can be done in many ways, I will list two of them.

6.1 Repeaterdev

One way can be to use the service repeaterdev (https://repeater.dev/). It can be used to send HTTP request to and endpoint and record the result. So we can just create an endpoint, run our code on a server (so it's running 24/7), and listen for daily requests from repeaterdev, which will also record the daily information that we can return in the endpoint. (We can use Flask or Django for creating server).

6.2 Cron

The other way is to create a cron job to schedule daily runs of our code. All unix operating system support this command line utility. So the information will be updated every day.

References

[1] *VADER-Sentiment-Analysis*. nltk. URL: `https://github.com/cjhutto/vaderSentiment`.

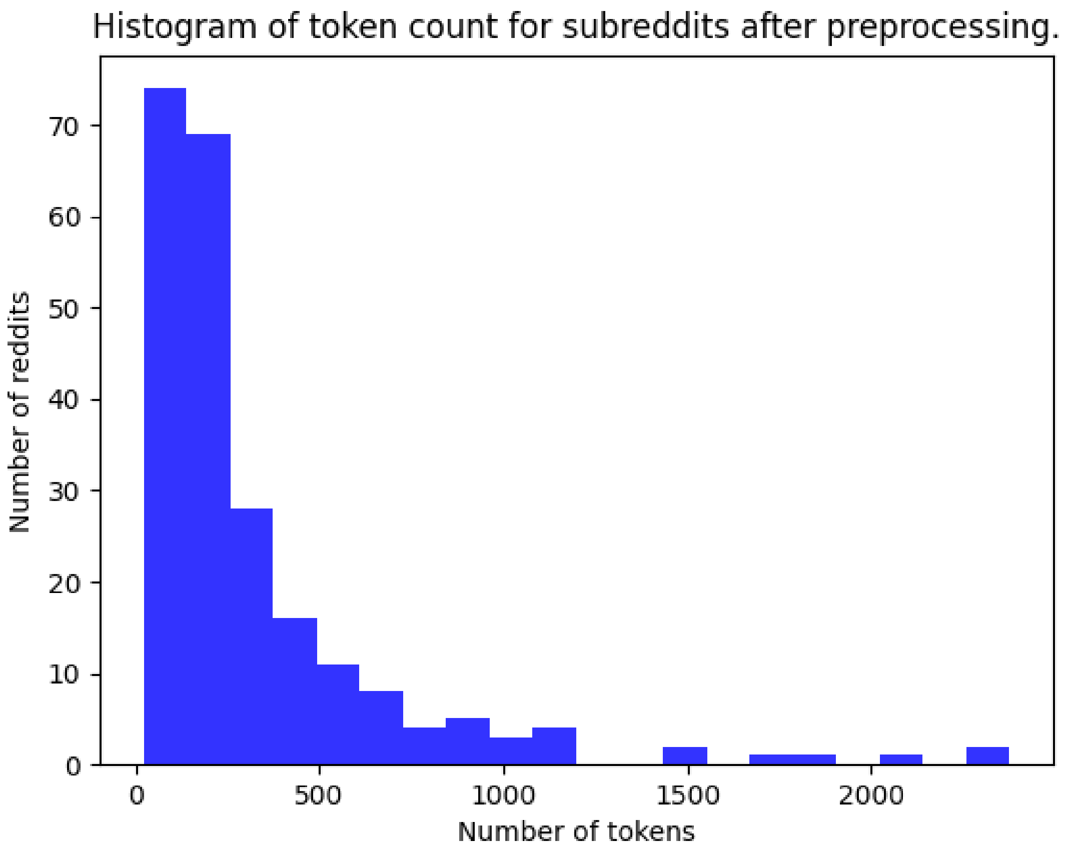

Figure 1: Word count

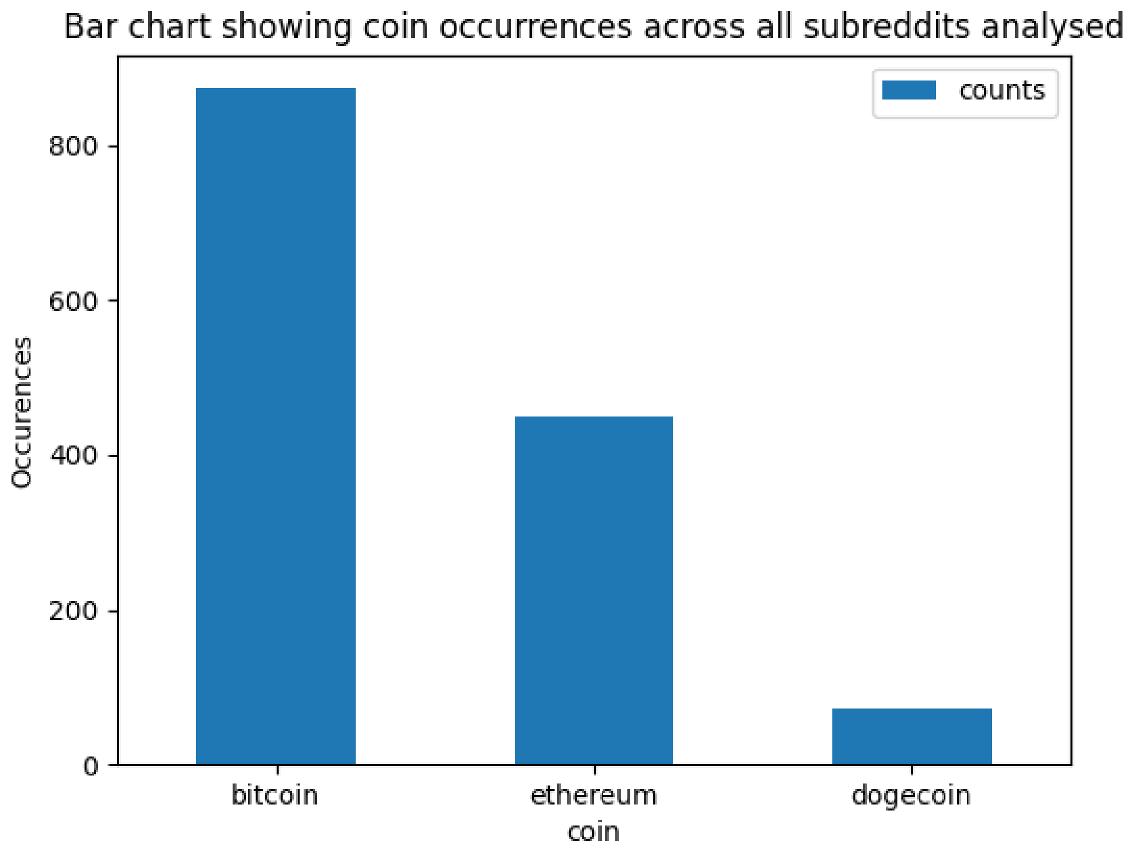

Figure 2: Coin occurrences

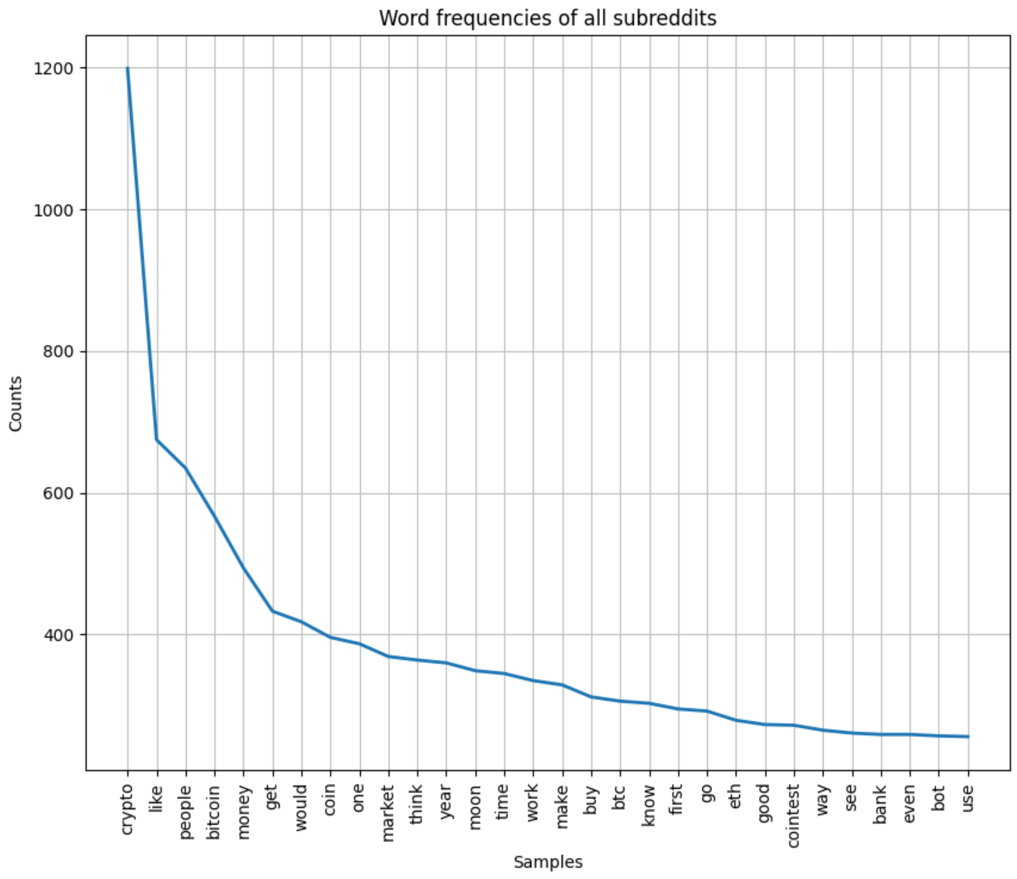

Figure 3: All word frequencies

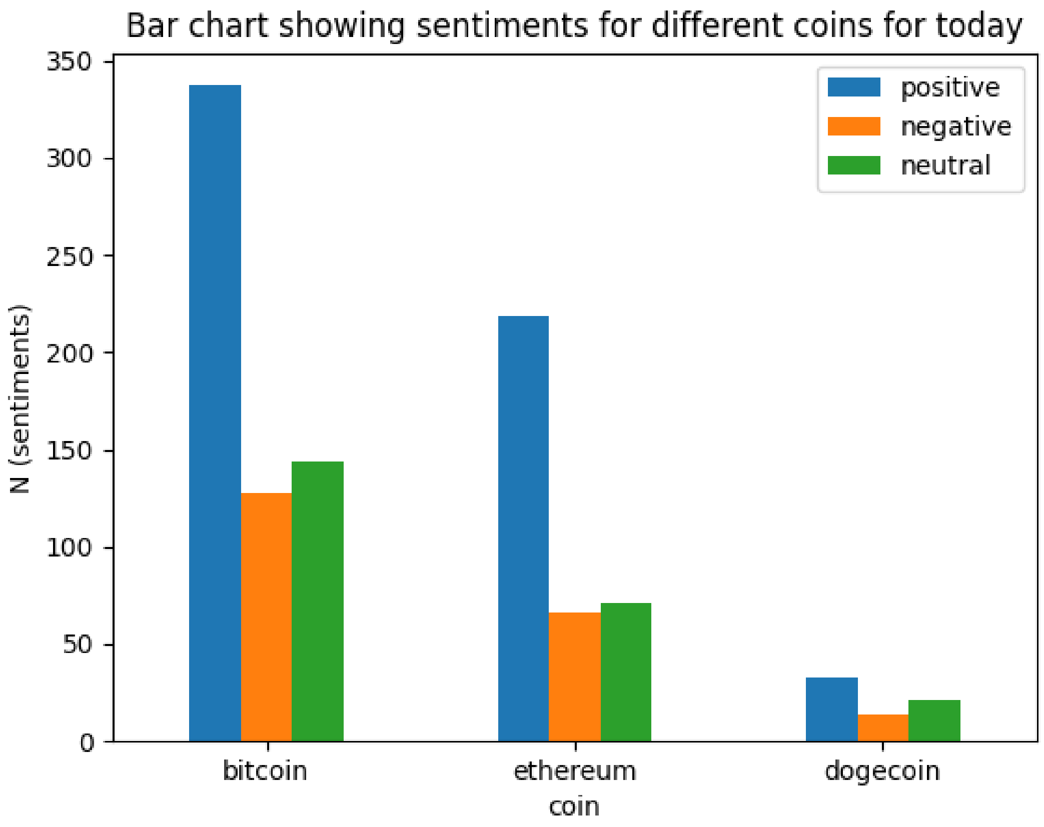

Figure 4: Sentiment analysis on all cryptocurrencies

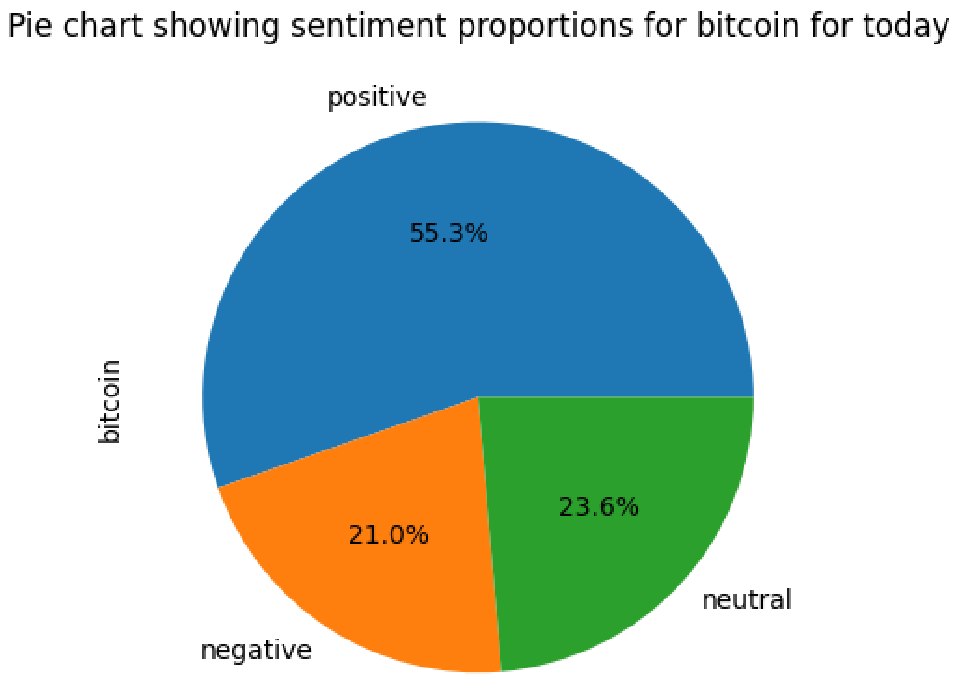

Figure 5: Pie chart for bitcoin

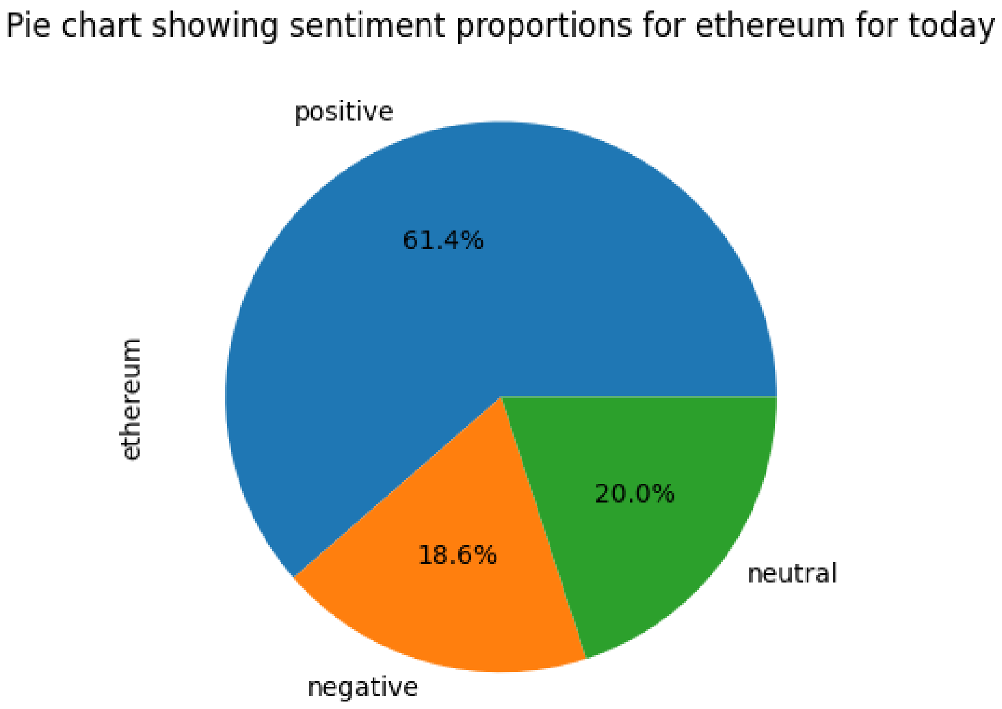

Figure 6: Pie chart for ethereum