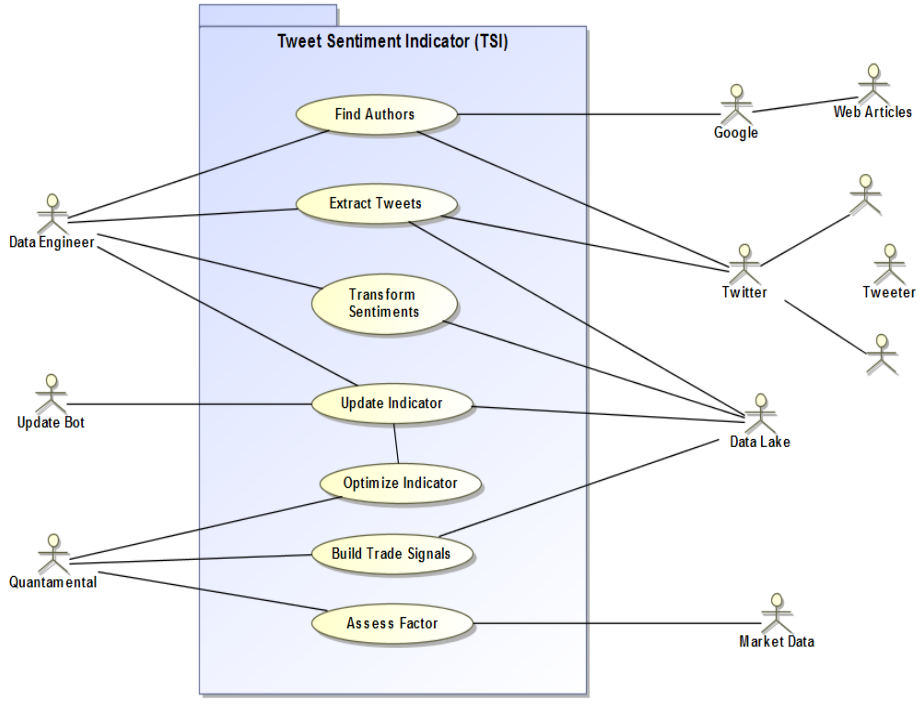
Udacity Data Engineering Capstone Project

Twitter Sentiment Indicator

In this project the aim is to extract Twitter tweets, transform them into sentiment scores, and load them as a timewise varying sentiment indicator intended for trading in the markets and ongoing machine learning developments.

Specifically, Twitter Sentiment related to cryptocurrencies is studied. It seems generally accepted that Bitcoin is the most well-known cryptocurrency. A search for Twitter accounts related to cryptocurrencies is made. The tweets from these accounts are mined, cleaned, and filtered for context related to Bitcoin. A Twitter Sentiment Indicator is built and implemented for comparison to Bitcoin market prices. A finance model applies trading rules to the indicator and generates factor trade signals. The finance model indicates some level of quality in the indicator.

Development Use Cases



The development use cases need flexibility in data handling that enables machine learning practices to be applied. This is because machine learning approaches can involve the generation of many models in the pursuit of optimal model.

A Quantamental or Quant role describes a mix of both Technical Quantitative and Fundamental elements involved in machine learned trading algorithms. Fundamental models can involve hearsay related to market status. For this project the hearsay comes in the form of a tweet that is converted to numerical values and quantitatively used for market study, enter the Quant.

The Quant is essentially the customer of the Data Engineer who is focused on loading useful data for the Quant who may use it in ensemble approaches combined with other factors in addition to side by side comparisons. The Quant models evolve to levels of refinement which lend confidence in trading methods.

The Data Lake is chosen as the practical solution to managing the data for this problem. The data can sit idle while new models are generated and compared to older versions. The flexible framework of the data lake affords emergent approaches not foreseen to be applied in future developments on a low data demand basis. Since the data is mostly idle, the data lake provides a low-cost approach to storage.

Find Authors Use Case

Justification: The twitter universe if very large and we are interested only in the twitter poets that likely influence the trading markets being monitored.
Story: The DE google searches for twitter accounts related to context of interest. Google returns results and the DE copy/pastes twitter handle “seeds” into a csv file. Using a Jupyter Notebook, the DE then downloads the csv of seeds and mines them for the Twitter handles of the accounts the seeds are following. The original seeds are combined with the following handles and count sorted for occurrence, the top 200 or so handles are selected as most popular for tweet mining, this list exported to a pickle file.
Acceptance Criteria: At least 50 seeds (Twitter account handles) are manually found via Google search. The following accounts of the seeds are found. Account handles intended to be mined are archived.

Extract Tweets Use Case

Justification: Only tweets from the specific twitter seeds of interest are mined or extracted for study. This reduces sentiment noise that is unrelated to the sentiment of interest.
Story: Using another Jupyter notebook, the DE downloads the archived seed list and mines tweets for each account over the last 10 years. The mined information is temporarily staged in a panda's data-frame before being exported to a pickle file to establish the raw data set.
Acceptance Criteria: over 1 million rows of tweets are found. Tweets are archived.

Transform Sentiment Use Case

Justification: People are given poetic license to tweet whatever they want, including NaN's, non-sentiment information like typos, url's and other handles.
Story: Using another Jupyter notebook, the DE loads the raw data set and assesses the data for Quality, including duplicate records, duplicate tweets, no records, corrupted language, non-English language, short tweets, @handles, urls, spelling typos. The tweets are cleaned using regex routines, spell checking and spell correction. Clean tweet data is exported using pickle.
Acceptance Criteria: Tweet defects are found and tagged.

Justification: It is conceivable that some twitter poets produce tweets that are not useful, possibly their vocabulary is limited or they contribute infrequently.
Story: The clean tweet data is uploaded using another Jupyter notebook, whereupon data analytics is used to identify outliers and remove them.
Acceptance Criteria: Bogus twitter poets are identified and tagged.

Justification: The words of the tweets are converted to a number that represents sentiment. The sentiment conversion may not work with all word combinations.
Story: Using another Jupyter notebook, the scrubbed data is converted to sentiment using an AI library or inference. The sentiment augments the tweet data-frame for further use downstream. Analytics is used to filter anomalous results from perpetuating further.
Acceptance Criteria: Each tweet is assigned a numerical sentiment score. Outliers are identified and tagged.

Justification: People have many different interests with different moods depending upon the subject. The idea is to filter the context by searching for keyword subjects within the tweets. The aim is to lower sentiment noise by filtering out tweets unrelated to the sentiment of interest.
Story: The tweet is tokenized into words. Common stop words are removed to create keywords. Keyword search related to context checks for presence or absence of keyword matches between a row and a keyword list of interest.
Acceptance Criteria: Each tweet is checked for context, outliers are tagged.

Update Indicator Use Case

Justification: Sentiment and price changes ongoing.
Story: an ETL script is run that updates the indicator to reflect current data.
Acceptance Criteria: An ETL script extracts recent tweets, transforms, and loads the new data onto old into a csv file for trading purposes.

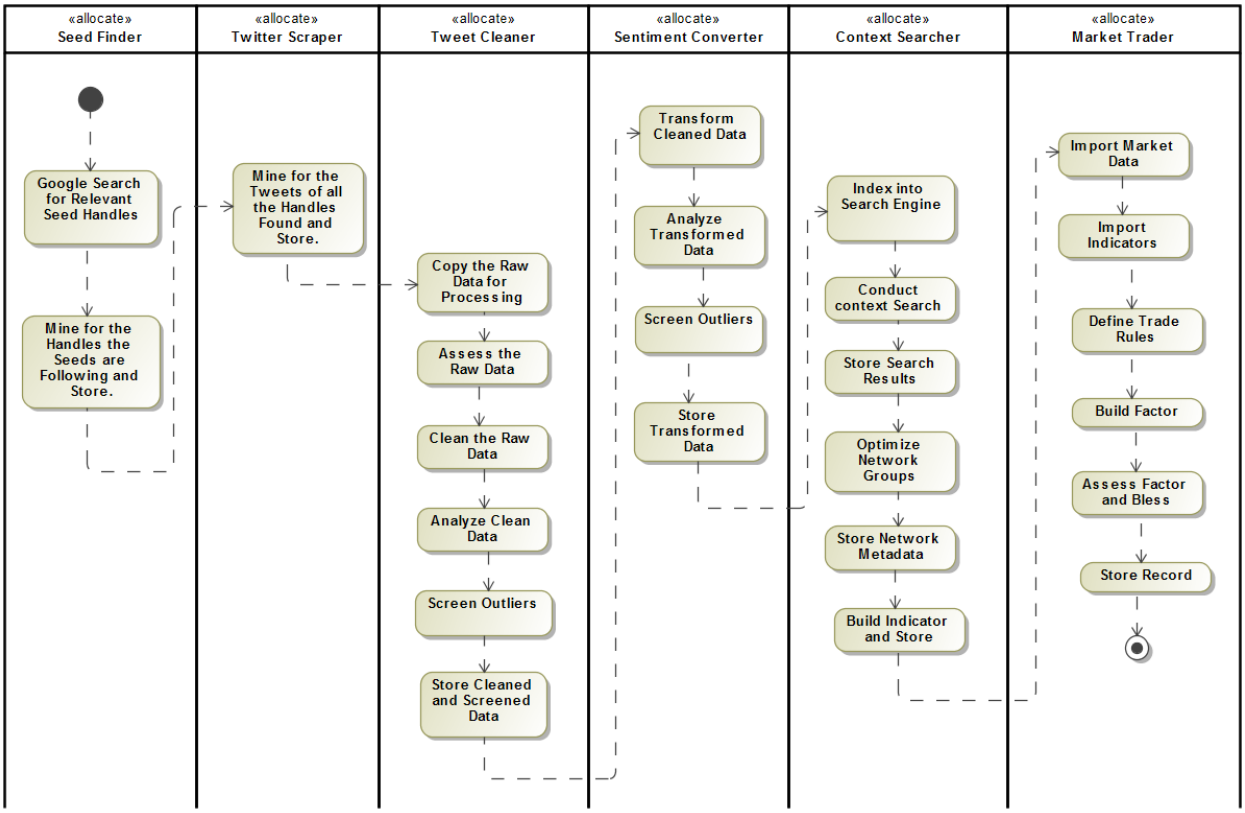
Build Trade Signal Use Case

Justification: The Indicator by itself is not useful until it is combined into a trading system. The indicator forms a trade signal using business logic devised to promote trading profit.
Story: The TSI is loaded into a finance model and integrated with market prices, and business logic to build Trade Signals.
Acceptance Criterion: Crude threshold crossing signals are created for testing.

Assess Factor Use Case

Justification: The utility of the indicator is gauged for value.
Story: The Trade Signals of the TSI get back-tested with market data to assess the quality of such a trading factor, i.e. to ascertain return value, baselined either to market performance (daily returns) or within and without an ensemble of indicator signals.
Acceptance Criterion: A t-test for factor returns and a Sharp Ratio are calculated for testing.

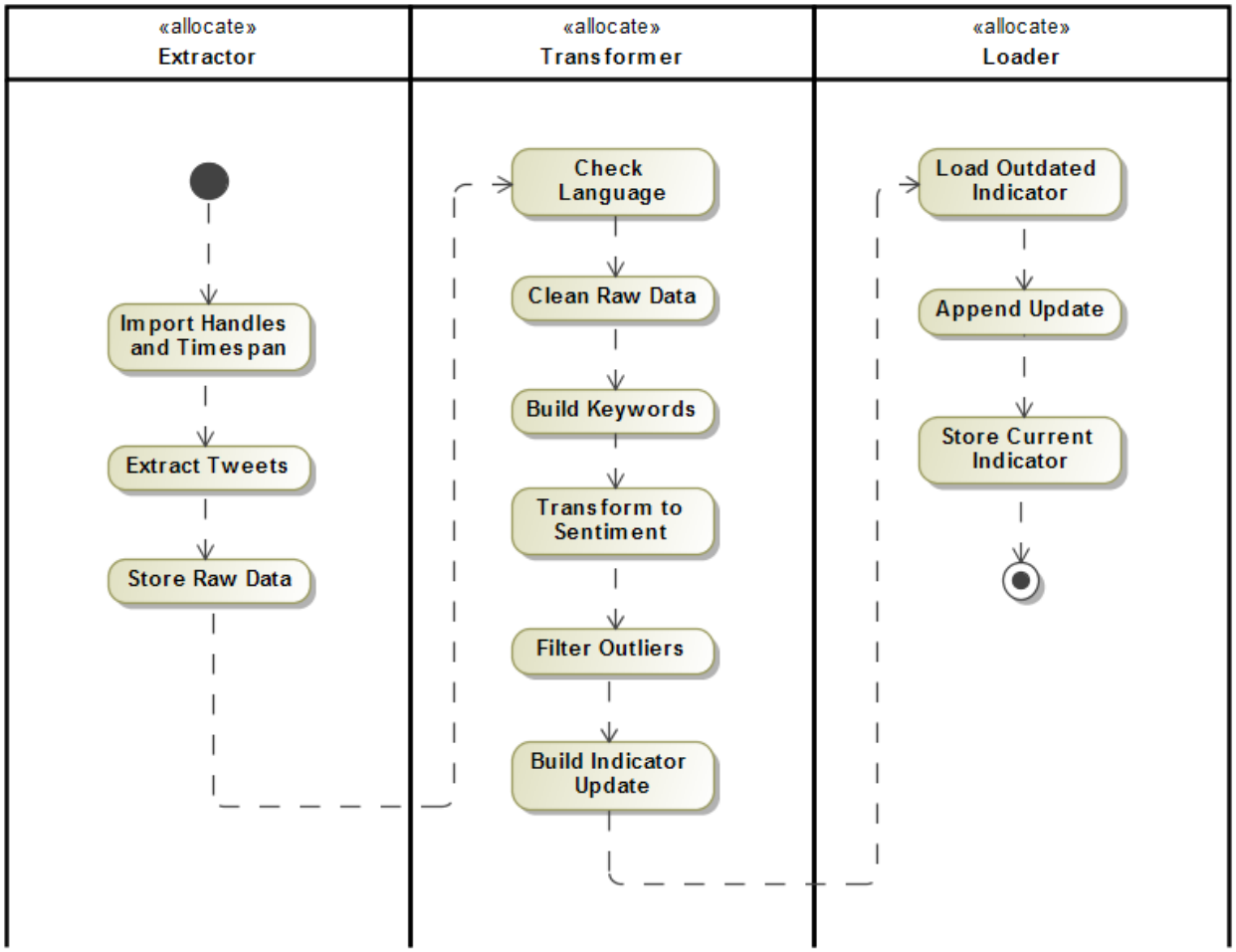
Development Activities by Jupyter Notebook Functions



The Indicator development in broken down into stages, where complexity is organized into notebooks that divide and conquer pieces of the whole pipeline.

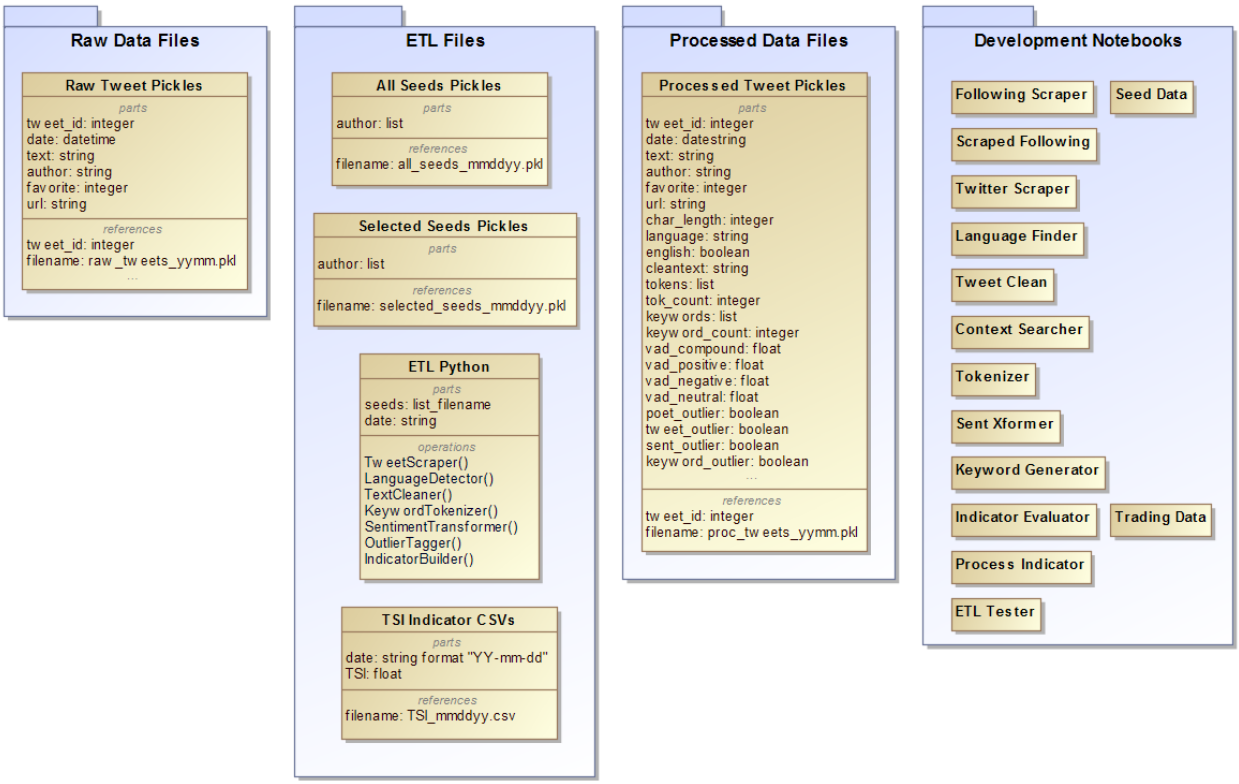
These stages involve finding the seed account names of relevant twitter poets, mining their tweets, cleaning their tweets, converting tweets to sentiment, filtering sentiment by context, and evaluating the indicator.

ETL Activities



This is the basic work flow of the ETL script, with significant portions adopted from the code in the development notebooks.

Data Model and Data Lake Structure



The packages could be imagined as folders holding data. What's in the folders are members of the data lake and are datatypes descriptions are noted. For simplicity, the documentation folder is left off. The raw file is carried through the ETL and augmented for machine learning purposes. More features may afford better opportunity to learn the data in ways beyond the scope of the Data Engineering's ETL. The pickle files are date partitioned by Month and Year. This is to afford the updates smaller files sizes and network speeds. The size of the raw data pickles is less than 15mb, the size of the processed pickles is less than 35mb. The size of the processed TSI is about 60kb. In whole the raw data occupies about 600mb, for 260 authors going back 10 years. There's about 2.5 million rows in the dataset. The variety of notebooks are used to develop small portions of the ETL processing before they are integrated to run all together. It's not difficult to imagine improved stages could be easily implemented using this flexible data lake structure.

Data Model Requirements

Number	Name	Text
1	ETL Function	Shall extract tweets from Twitter, transform them into sentiment, and load them as Twitter Sentiment Indicator (TSI).
2	ETL Updates	Shall be able to update TSI on daily basis. Note, this is intended to run in the middle of the night for Tweets made the day before.
3	TSI Time Scale	TSI shall be aggregated over days for comparison with daily market prices. Note, round dates to nearest noon.
4	TSI Sentiment Scale	TSI sentiment level shall be normalized to span from 0 to 1. Note, this is for use with business rules containing fixed threshold levels.
5	Data Lake Storage	Project shall be stored in data-lake. Note this is to accommodate ongoing development within processing stages and daily production updates.
6	File Types	File storage shall consist of Pickle and CSV compatible with non-secure public accessible use with Pandas and Jupyter notebooks.
7	Tweeter Finding	Tweeter user names (seeds) shall start with manual search followed by Twitter mining of seed account friends (accounts seeds are following.)
8	Tweet Extraction	Tweets shall be extracted from Twitter.
9	Language Detection	Tweet language shall be detected on a best effort basis.
10	English Filtering	English language shall be indicated with boolean field.
11	Declutter Cleaning	Cleaning shall include regular expression clutter removal. Note, this is intended to reduce noise from Sentiment converter trained on English, i.e. urls, pics, usernames.
12	Author Screening	Author screening afforded using selected authors file.
13	Author Filtering	Acceptable authors shall be indicated with boolean field.
14	Sentiment Transformation	Tweets shall be transformed into numerical floating point sentiment. Note, intended to provide a continuous function varying between 0 and 1.
15	Sentiment Filtering	Acceptable sentiment shall be indicated with boolean field.
16	Keyword Generation	Keywords shall be generated as Tweet word tokens filtered free of stop words.
17	Keyword Filtering	Acceptable keywords shall be indicated with boolean field.
18	Tweet Filtering	Acceptable tweets shall be indicated with boolean field. For example, tweets less than 3 words shall be tagged as not okay.
19	Indicator Generation	TSI shall consist of the daily average of tweet sentiment scores that occurred within acceptable filter indications.
20	ETL Running	ETL shall be a Python script that can run from terminal. Note, this is distinct from Jupyter notebooks used for development.
21	Trade Signal Build	Trade Signals (1/0's for in/out) shall be built using business rules together with TSI indicator. Note, simple rules consist of threshold crossovers and time shifts expected.
22	Factor Assessment	Trade Signals shall be back-tested against market prices of interest. Note, this is likely an ensemble of indicators with and without a TSI factor.
23	Write-Up	Project write-up shall follow project guidance.
24	GitHub	Project shall be uploaded to GitHub.

Step 2: Extract the Data

Original Twitter Seeds

Two websites were returned by Google Search of "twitter crypto accounts going back to 2009":

1) "19 Bitcoin Accounts You Should Follow on Twitter", by Jeff John Roberts and David Z. Morris, 12-27-17; http://fortune.com/2017/12/27/bitcoin-twitter/
2) "The Top 100 Crypto and Blockchain Influencers", by Leon Chevalier, 2-28-19; https://medium.com/blockinfluence/the-top-100-crypto-and-blockchain-influencers-on-crypto-twitter-bfd33668e010

From these two websites, 51 Twitter Account Handles were manually copy/pasted into "crypto_seed.csv".

Mining of the Accounts Names that Seeds are Following

The python Tweepy API was used to find out who the Seeds are following.

The Jupyter notebook "tweet_followers_r2.ipynb" is used to scrape Twitter for accounts. Since Tweepy has a rate limit, for each seed account a pickle output file is built to buffer timeout problems. Some accounts take several hours to mine. There's 51 files of format "following##.data", these are pickles of author screen names, they range from 1kb to 280kb in size.

Assessing the Mined Account Names

The 51 pickle files saved for the seed followings are imported into the Jupyter notebook "scraped_following_r1". Here, it's revealed that there's 71,679 accounts followed in total but they are not necessarily unique. These accounts are counted and sorted to find the most common. Together with the original seeds, a list of 264 account most common handles are pickled to a file for tweet mining,

"seeds_061419.csv"; contains the author screen names and id's, 6kb in size.

"crypto_tweeter_seeds_060419.pkl" another constructor, 5kb in size.

Mining Twitter for Tweets

A Jupyter Notebook called “twitter_scraper_r2.ipynb” is used to import the account names and mine the tweets over 10 years. Note that since the mining is so time-consuming due to network and data limits, that both twitter account and twitter tweet mining was happening using different notebooks. When all is said and done, 2,563,802 tweets are mined and store as separate rows of a panda’s data-frame before landing in a pickle file as raw data, this pickle file is 600mb in size:

‘tweety_060519.pkl’ ; contains 2.5M raw data tweets, 605mb in size.

The big file is Date Partitioned in a Raw data folder, there’s about 127 files of format “raw_tweets_yyyy_mm.pkl”.

For updating the latest year month file is downloaded, appended, and re-uploaded.

Step 3: Explore and Assess the Data

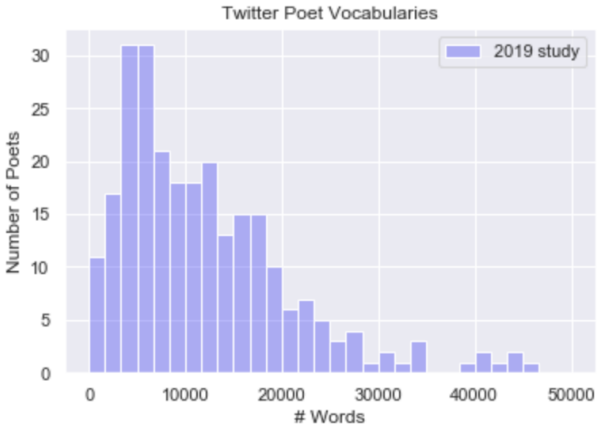
Looking at Language

Google Language Detector, a python library, is used to tag tweets for the type of language the tweet is made in. Starting with 2.5M total tweets, this drops the tweet count down to 2.3M for English only, about 10% detected as not English. This is considered a noise source for a sentiment transformer trained on English.

This Language detector is very slow when run on one core. It would have taken three days to run through the big dataset. For this reason, the data-frame was sliced in smaller chunks and 8-core Multi-Processing was used to finish the job in a little more than an hour. A fallback “unknown” tag was used for cases that the language couldn’t be detected.

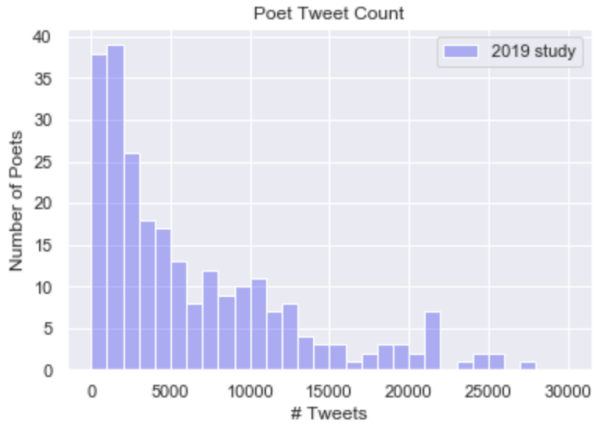
Looking at Poets

Words get tokenized and counted to get a sense of people's vocabularies.



It's not difficult to imagine some poets have poor spelling and others just don't know that many words. If we draw the line at 30,000 words between okay poets and not – then the account "realDonaldTrump" is unfortunately excluded as having a 34,583 word vocabulary. We did not do spell checking for typos.

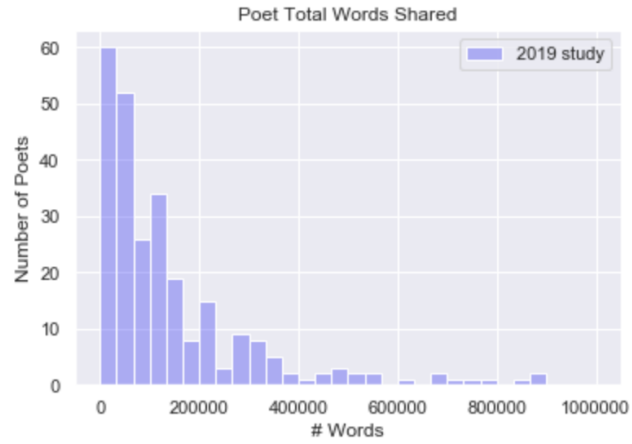
Furthermore, the poet tweet count was plotted.




```
mined dataset
      kurtosis= 13.8    skewness= 2.8      sum= 3344991
```

The English tweet count range spanned from 18 to 174,184 in 10 years.

Likewise, from the total words tweeted we compared poets.




```
mined dataset
      kurtosis= 13.8    skewness= 2.8      sum= 3344991
```

The range of the total words tweeted spanned from 400 to 2.4M over 10 years.

Together, the poet acceptance thresholds were set:

Okay: (vocab < 30,000 words) & (tweets>3000) & (total words < 1M).

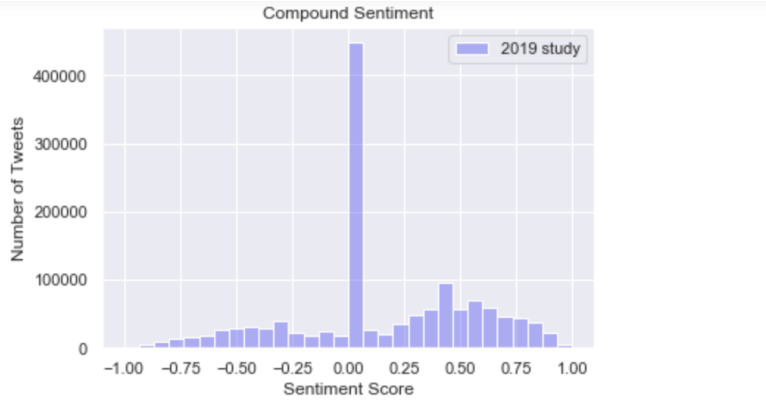
This dropped the number of okay authors to 143 out of 262 we started with.

These are contained in the “screened_seeds_061419.pkl” file used for tagging.

Starting with 2.3M tweets after English screening, the poet screening drops the tweet count down to 1.4M.

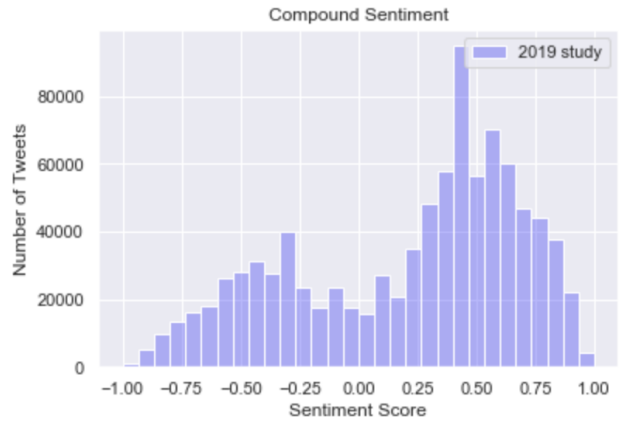
Looking at Sentiment

The tweets are converted to sentiment using the Vader Sentiment Intensity Analyzer. This reveals four sentiment scores: positive, negative, neutral, and compound.




```
mined dataset
      kurtosis= -0.4    skewness= -0.2      sum= 196302.51609999995
```

The large spike at zero seems out of the ordinary and complements a neutral score of 1 in those cases. These are interpreted as a Sentiment Gauge failure, and tagged as not okay. Here, Okay: (vader compound != 0) & (vad neutral != 1).




```
mined dataset
      kurtosis= -0.8    skewness= -0.6      sum= 196302.5161
```

The filtered result is a more normal looking bi-modal distribution.

This drops the tweet count down from 1.4M to 940K.

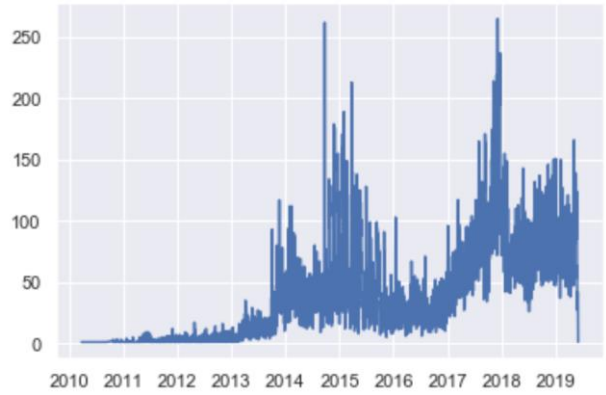
Looking at Keywords

The tweets are tokenized, lower-cased, and common language stop-words are removed. The resulting keywords are checked for presence or absence of context keywords.

In this case, the context keywords chosen as: (bitcoin, #bitcoin, btc, bitcoins).

This further reduced the dataset, from 940K down to 127K tweets over 10 years.

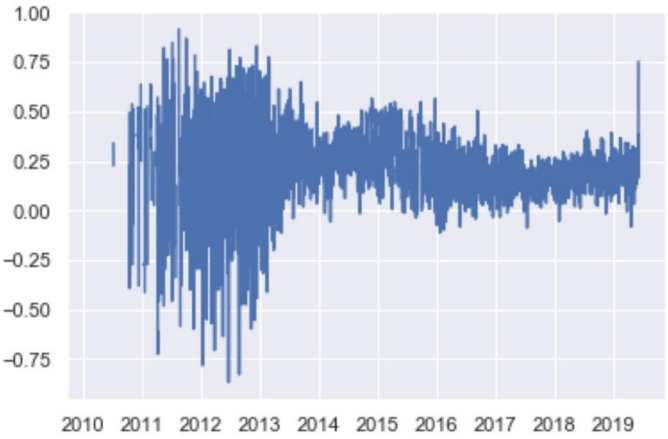
A plot of this tweet count over time:



Looking at Time

Unfortunately gap days are bad news for the indicator. NaN's days are to be avoided. The volume of tweets shifts around 2014. The time cut-off is chosen as the start of 2014. In this case, the compound sentiment values can be observed as having reasonably consistent behavior as compared with prior.

Filtered Tweets: Compound Sentiment vs. Time

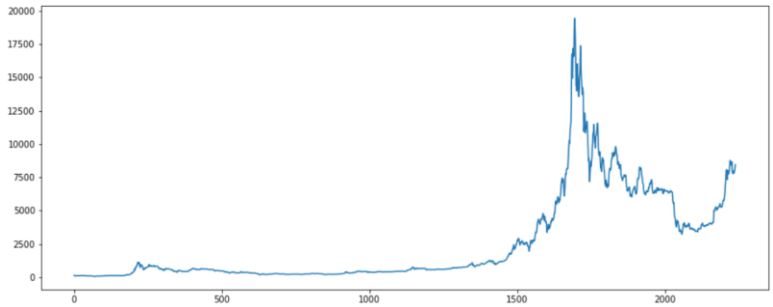


A slice of this mean sentiment aggregated over days reveals the TSI indicator.

Looking at Price

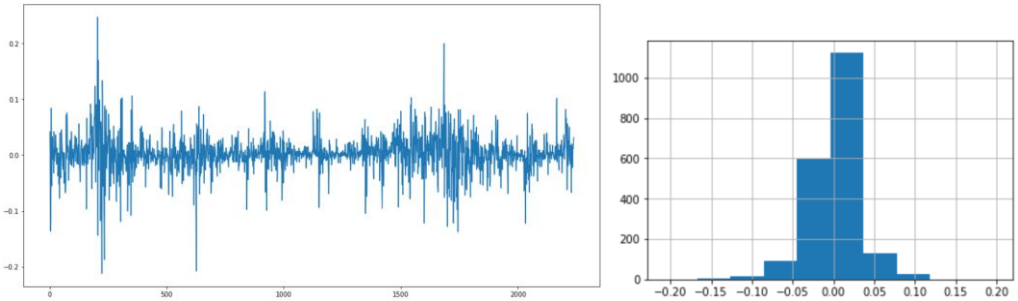
The Bitcoin price in $USD vs time plot seems infamous.

BTC in $USD, 2013 to 2019



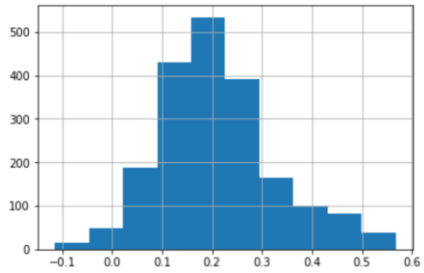
The Bitcoin daily return seems less common:

BTC daily log returns, 2013 to 2019



The Daily Indicator Sentiment seems well behaved, slightly biased optimistic:

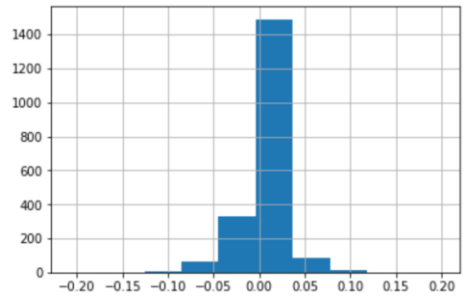
The TSI distribution from 2014 onward.



Using crude contrarian logic as trading rules where action is taken when the indicator crosses thresholds, the trading factor is formed.

For example, if we are out of the market and looking to enter, we enter when the Indicator is less than one. We hold that position and look for an exit threshold when it crosses greater than 0.3 and so on, ongoing.

Looking at the returns from such a strategy, they peak non-zero.



A T-Test with null hypothesis of zero mean using standard deviation of the distribution, results in a large t and miniscule p; this indicates significance where we can reject the null hypothesis.

Expected returns over 5 years is estimated at ~736%.

A Sharp ratio over the same period is estimated at 3.8

In summary, the indicator seems to work okay for educational purposes.

(It is not recommended to buy bitcoin using this study.)

Step 4: Run ETL to Model the Data

ETL Test

After development modeling, an ETL process notebook was built to roughly piece all the ETL steps together, shy of the following and tweet scraping. Once a reasonable flow was attained, the notebook was transcribed into Python files.

The ETL augments the raw data, passing a data-frame of growing size from one stage to another. In addition to passing the data-frame along, the stages commonly pickle it to file. Some of the stages are time consuming and if some unforeseen error is encountered, backstep processing is reduced to a file load.

The 7 distinct stages include scraping tweets from Twitter, checking the language, cleaning the text, tokenizing keywords, transforming text to sentiment, tagging the outliers, and updating the Twitter Sentiment Indicator (TSI). Each of these stages is organized in a separate Python file that is called by the ETL script. The ETL script was developed using the ETL Test worksheet. Here each of the stages was tested and progressed to the next stage until ultimately at the bottom of the notebook, the ETL.py script was run in its entirety without errors. This script successfully updates but it is intended to be run only as needed for educational purposes only.

Each stage in this ETL can be time consuming. Mining 2.5M historical tweets alone takes several days, other steps together can take many hours.

Step5: Conclusion

Requirement Review

Most of the requirements reached for in the project scope are functional in nature and were attained. The next step may be assigning performance requirements that push it to the next level. For example, maybe we want some Sharp Ratios > 6, or return > 1000%? Enter machine learning optimization.

The greater purpose of the early functional requirements may be the focus they demand in simply formulating and following through on implementing basic functions. Here the Data Quality is mostly managed using tag approaches that afford future Machine Learning projects the option of using them or not. The fact that raw data is augmented allows alternative implementations to be tested side by side for improvement comparison. The utility of large pandas data-frames in this respect also promotes greater numbers of fields to train on, for purposes other than Sentiment Indicators alone.

Summary

In this Data Engineering project, an ETL process was developed to extract Tweets, transform them to numerical Sentiment scores, and load them into a Twitter Sentiment Indicator intended for financial study. Specifically, cryptocurrency related tweets were downloaded in mass and filtered down in relevance and behavior to produce what appears to be a financially functional result. Left to its own devices this may be too good to be true. Optimizing its hyper-parameters and incorporating this into an ensemble of indicators seems obvious follow on projects for further study which may ground it with sound profitability potential.

This project seems the fruit of three Udacity programs: Natural Language Processing, AI for Trading, and Data Engineering. Thanks for that.