**Intro:**

Sentiment analysis is a major topic in today’s financial markets, especially how sentiment interacts with different types of financial securities. Often, sentiment can be a leading indicator for a bull or bear market and an investor can create alpha with the correct sentiment indicator. However, Twitter sentiment is notoriously complex for a number of reasons, folks speak in slang, are sarcastic, and make spelling and grammatical errors. All of this complicates a machine learning sentiment analysis task, to predict sentiment in tweets.

While these complicating factors can present issues, a lot of the noise can be limited by honing in on a financial sector like structured finance. The analysis further delves into a subsector within structured finance called Commercial Mortgage-Backed Securities (CMBS) and related twitter accounts. The tweets pulled will be a mixture of official twitter accounts and individuals tweeting about structured finance. One final note, the code was created for people within my department that have limited to no experience in python. Generally, most are downloading for the first time, so creation of many functions was utilized to help simplify the coding process.

**Data and Analysis:**

To build the sentiment analyzer, a random assortment of tweets were necessary to pull. In order to pull old tweets, the “GetOldTweets3” package was utilized. TwitterCriteria, was the function utilized that required a collection of search parameters to be used together with TweetManager.

* setUsername (str or iterable): An optional specific username(s) from a twitter account (with or without "@").
* setSince (str. "yyyy-mm-dd"): A lower bound date (UTC) to restrict search.
* setUntil (str. "yyyy-mm-dd"): An upper bound date (not included) to restrict search.
* setQuerySearch (str): A query text to be matched.
* setTopTweets (bool): If True only the Top Tweets will be retrieved.
* setNear(str): A reference location area from where tweets were generated.
* setWithin (str): A distance radius from "near" location (e.g. 15mi).
* setMaxTweets (int): The maximum number of tweets to be retrieved. If this number is unsetted or lower than 1 all possible tweets will be retrieved.

The following types of data could be pulled:

* id (str)
* permalink (str)
* username (str)
* to (str)
* text (str)
* date (datetime) in UTC
* retweets (int)
* favorites (int)
* mentions (str)

Also notable was geography was a type of data that can be pulled, however, since Twitter updated there terms, the geography field was not populated with any data when pulled.

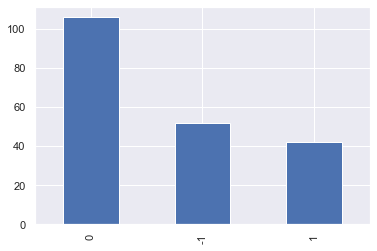
Lines 24-169 showed the process of how I wanted to build out the sentiment analyzer. As I searched different key words, I began realizing even something as specific as CMBS, didn’t return the results I had expected. In fact, there were a number of professional European Football tweets that were related to an account called “CMBS\_football”. So, I switched gears slightly and began looking for CMBS twitter accounts that were either news related or a strong individual voice in the market. Also, a number of my colleagues work in a subsect of structured finance called “ABS” and I didn’t want to return 1000’s of tweets related to “how I lost my washboard abs during the pandemic”.

Functions:

* tweets\_by\_practice:
  + The function requires three inputs:
    - Group = structured finance subsector
    - Date\_start = the start date a user wants to pull from, must be entered in “mm/dd/yyyy” format
    - end\_start = the start date a user wants to pull from, must be entered in “mm/dd/yyyy” format
  + The file path looks strange due to it being a shared drive within a company folder, however this is where interesting twitter accounts are stored called “SF\_twitter\_accs.xlsx” is stored.
  + Then loops through each twitter account listed by structured finance subsector and then pulls all information related to the tweet. Finally it converts to a pandas dataframe and adds columns for users to eventually update their twitter sentiment.
* Save\_tweets:
  + This saves down a random sample of tweets that the user specifies via a number input parameter.
  + It also saves down a copy of the tweets pulled, in case there is a crash or someone loses there tweets. They can come back.
  + Also the return, also returns a file path variable. So that once the sample is updated with tweets, the user can just read in the same file once sentiment is assigned to tweets.

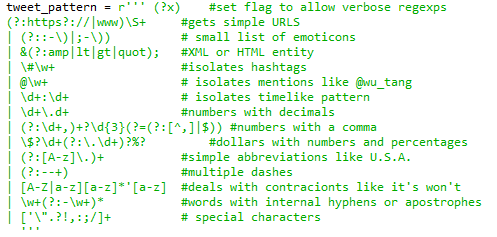
This was where I asked users to stop, and assign tweets in the file that was automatically saved down.

A quick look at the sentiment that I assigned from the random sample of tweets sourced from 6/1/2019 to 6/1/2020:



The bar plot shows that a majority of the tweets I classified as neutral, and the overall sentiment on average was slightly negative since I classified more tweets as negative than positive.

The next piece was to begin the NLP process, tokenization occurred by two different means, one removing stopped words, and one keeping stopped words. Next any links in the tweets were removed (this was an addition, after the first iteration of the Naïve Bayes algorithm was returning poor results). Then the tokenization utilized was based on the following regex code:



The main reason for using the regex code was to keep multiple hashtags as their own token, multiple punctuation together, dollar signs, and contractors. The NLTK tokenization pattern would not allow for these tokens to be combined, so I refined.

The next piece was to slightly tweak the featuresets and POS features provided in Lab 9. These were looped into the “get\_all\_WF\_unigram” function. The function does the following:

* Takes the sample dataframe, isolates the “Majority” column and creates a new column that assigns strings to each value.
* Next it creates four different columns in the dataframe and tokenizes 4 different ways:
  + Unigram
  + Unigram with stop word removal
  + Bigram
  + Bigram with stop word removal
* The function then takes each and creates featuresets, the number of matched featuresets are input by the user in the parameter information. Also the function creates part of speech featuresets as well.
* Once featuresets are assigned, there are four print statements that indicate to the user how large a sample they selected out of the universe of words.
* The function then returns the five featuresets and also word features for both non-stopped and stopped varieties. This is important for the final step when taking the remaining universe of tweets and predicting sentiment.

The function “NB\_class\_all” applies the Naïve Bayes algorithm to each class of featuresets and prints a score for each.

The last function then applies a user specified classifier task to the remaining tweets. This appends the sentiment score to the dataframe passed through so further analysis can be utilized.

**Results**

Running through the classifier, the overall accuracy for each type are shown below:

* Accuracy of unigram 0.60625%
* Accuracy of stopped unigram 0.6%
* Accuracy of bigram 0.60625%
* Accuracy of stopped bigram 0.6%
* Accuracy of POS tags 0.59375%

The following were the top 30 word features included in the classifier:

Most Informative Unigram Features

The most informative issues will definitely need to be refined as a number of items like “real” or “estate” would be more neutral than positive. However, there are some interesting tidbits. V\_#multifamily showing negative is interesting. Multifamily = apartment buildings, or large block rentals together. It’s very interesting that multifamily is showing negative, as this is something that has been swept under the rug in terms of talking points within the CMBS market.

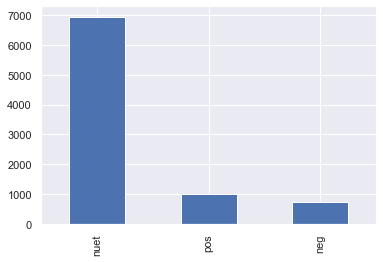
Further information surrounding this development is warranted, and looking through more data will help understand if this is just a topic, or if something is starting to turn negative in the multifamily market.

Also notable, are a lot of stop words are being classified as positive or negative. Will need to further refine with a larger sample size.

Since the accuracy hovered somewhere between 55% and 65%, a few strategies to further bolster a stronger sentiment classifier are needed. The main issue is the sample size of only 200 tweets and it is likely more tweets need to be sampled. I plan to utilize the sentiment tags from the other subsectors within structured finance that will be classified in a meeting the week of June 22nd. This will broaden the spectrum of tweets sampled and yield approximately 1000 tweets that will be classified. Hopefully this will boost accuracy scores slightly, however, if this still shows lower accuracy scores, I plan to utilize pre-classified tweets mentioned here:

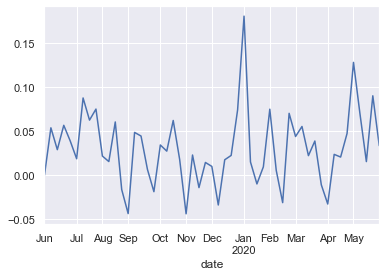
<https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-program-in-python-with-naive-bayes-classification-672e5589a7ed>

A quick synopsis of overall predicted sentiment:

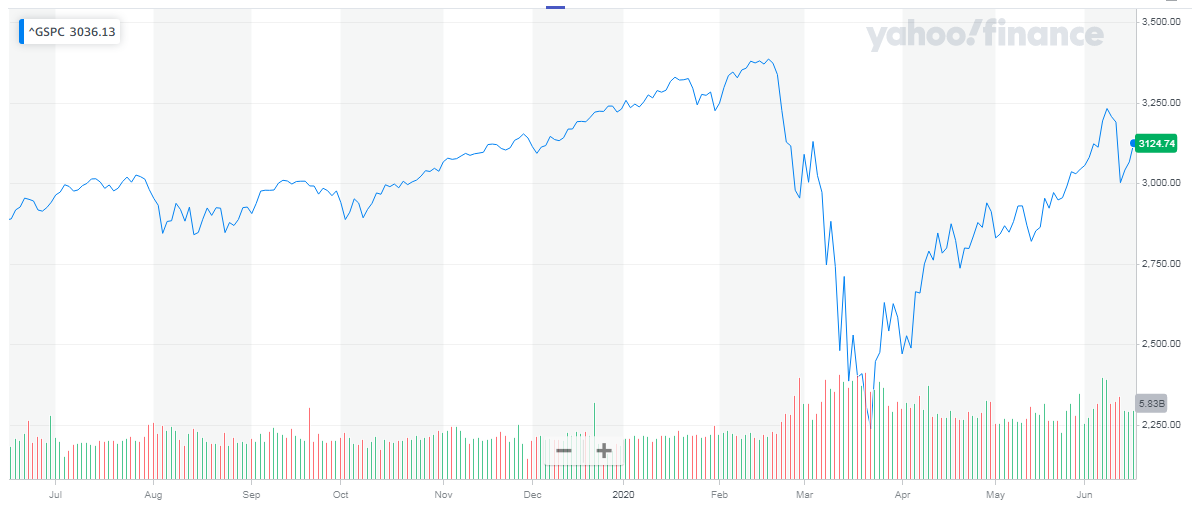


It’s likely a larger sample size and better accuracy will show more tweets in the positive or negative segments.

Sentiment analysis by week.



Vs the S&P 500:



Interesting, the massive drop off in the S&P 500 was preluded by a low in sentiment in mid-February. It also very interesting that positive sentiment has spiked at the beginning of May, coinciding with a pretty strong month.

**Conclusions**:

The results are promising and show potential for a leading indicator in stock movement. Some early results even seem to follow the trend of perceived sentiment, with lower sentiment in late February as Covid-19 was becoming problematic. And then dropped in early April as massive shut downs were coming into effect.

A very interesting aspect of the sentiment analysis was positive sentiment spiked in May. Does this mean that economy and broader sentiment is on the mend or is it a false hope? Time will tell, but for now more information never hurts when making decisions.