A drawing of a cartoon character

Description automatically generated

Kent Appiah, Seán Hayes & Jeff Levesque

https://github.com/jeff1evesque/ist-736 | Final Project

IST-736: Market Sentiment

PRofessor gates

Table of Contents

[Introduction 2](#_Toc13386812)

[Analysis 2](#_Toc13386813)

[Data Preparation 3](#_Toc13386814)

[Twitter API 3](#_Toc13386815)

[Quandl API 4](#_Toc13386816)

[Joining Data 5](#_Toc13386817)

[Exploratory 5](#_Toc13386818)

[Stop Words 5](#_Toc13386819)

[Topic Model 6](#_Toc13386820)

[Latent Dirichlet Allocation 6](#_Toc13386821)

[Selected Topics 8](#_Toc13386822)

[Sentiment Analysis 10](#_Toc13386823)

[Baseline Results 12](#_Toc13386824)

[Time series 12](#_Toc13386825)

[Granger Causality 12](#_Toc13386826)

[Classification 12](#_Toc13386827)

[Conclusions 12](#_Toc13386828)

# Introduction

Do the opinions of Financial Analysts on Twitter impact market sentiment and volatility? There are many financial analysts on twitter offering investment advice to millions of people daily. These analysts also have other mediums outside of Twitter, including TV shows, blogs and newspaper columns. However, Twitter can provide twitter sentiment towards the market that analysts are portraying daily. For example, Jim Cramer is arguably the most famous financial analyst today, having over 1.1 million followers on Twitter, and host of the popular Mad Money television series. While audience members often look to financial analysts such as Jim Cramer, one may question whether their advice impacts the investment decisions of followers, and in turn, impacting market volatility?

Market volatility is a statistical measure of dispersion relative to security or market index. Generally, higher volatility is a sign of greater risk since the range of possible values vary relatively more[[1]](#footnote-1). Market volatility can be tracked by using the CBOE Volatility Index (VIX). The VIX is a market index that represents the market's expectation of 30-day forward-looking volatility. It is derived from the price inputs of the S&P 500 index options; it provides a measure of market risk and investors' sentiments. The VIX is considered a reflection of investor sentiment; the higher the VIX is up, the higher the levels of investor anxiety and market volatility.

If there is a correlation between the sentiment being portrayed by these analysts towards the market and the level of market volatility for that day, then this information can be a valuable tool for investors to incorporate into their investment strategy. Having advanced knowledge of when a market is about to increase in volatility would be invaluable to investors allowing them to adjust their portfolios preempting the market risk. Tweets have been known to impact the market before, on May 5th 2019, President Trump tweeted negatively about a Trade War with China and the VIX rose by as much as 46.1% intra-day the next market day, while the Dow Jones Industrial Average plunged by as much as 471 points .

# Data Preparation

## Twitter API

The Twitter API[[2]](#footnote-2) was implemented using an approved Twitter developer[[3]](#footnote-3) account. A python template file (config--TEMPLATE.py) was created containing dummy text representing the secret key and tokens provided with the Twitter developer account. This file was copied as config.py, with values properly substituted. Changes were a requirement of the general application implementing the Twython[[4]](#footnote-4) package. Within the application, two main twitter functionalities were streamlined. The first allowed general querying through a set of parameters[[5]](#footnote-5), while the second allowed querying content for specified twitter screen names. This functionality was provided using the user timeline component[[6]](#footnote-6).

Five screen names were queried:

* Jimcramer
* ReformedBroker
* TheStalwart
* LizAnnSonders
* SJosephBurns

Corresponding code generated dataframe structures, for each of the above screen names, then outputted to an associated csv file[[7]](#footnote-7). On future executions, if the corresponding csv file already exists, then the twitter api did not duplicate exiting files.

Furthermore, the parameters collected from the twitter accounts were screen\_name, created\_at, and full\_text. Each account was collected using a rate\_limit=900[[8]](#footnote-8). This ensured that the maximum number of tweets could be collected per screen name. However, due to the request limit, roughly 15 minutes

needed to transpire before re-executing, to obtain the maximum content for the successive screen name. Thus, a little over 1.5 hours was required to initially generate local csv files.

Finally, a default start\_date = datetime(3000, 12, 25) and end\_date = datetime(1000, 12, 25) was defined. This definition was created to represent the datetime range for a given twitter screen name. Specifically, the initial start\_date was compared to each tweet for a given user. If a tweet exists with an earlier datetime, this was set as the new start\_date. This type of logic was extended similarly for the end\_date. This maximized value allows the functional tweet domain to accurately map to the quandl historical range.

## Quandl API

Like the Twitter API, the python Quandl API[[9]](#footnote-9) was utilized to acquire market data, including the Nasdaq index. An account was needed to obtain the associated API key, and the same config.py was utilized, respectively. Moreover, the date range was maximized[[10]](#footnote-10) in order to obtain the largest possible dataset. While obtaining data was not as restricted by the same rate limit as Twitter, a local csv file was created. This ensures integrity and optimization in case a future study extends with additional datasets. While five different columns were returned, only the Index Value was utilized for successive calculations, described in sections below:

* Trade Date
* Index Value
* High
* Low
* Total Market Value
* Dividend Market Value

Finally, as described earlier, market data range, was predicated on the maximized tweet domain.

|  |
| --- |
|  |
| Figure 1: domain mapping from Twitter API (x) and Quandl Data f(x)[[11]](#footnote-11). |

## Joining Data

To simplify processing, tweets were aggregated by created\_at and screen\_name. If an account tweeted multiple times a given day, each full\_text instance was concatenated to a single string. This allowed sentiment measure to be computed as a time series. Furthermore, each twitter account data was merged on Trade Date = created\_at column. Moreover, later classification tasks were predicated on comparing the current day market value with the previous day. Thus, some edge cases needed to be considered:

1. If the first Index Value is nan, drop the instance
2. If successive (n+1) index has a previous step nan, skip and do nothing
3. If successive (n+1) index is nan, set market values to previous day and concatenate current full\_text with previous day.

Additionally, if a given day contained an empty string for full\_text, this instance was dropped, and the dataframe index was reset.

# Exploratory

Initial exploration was performed for each twitter screen name, and overall aggregation. Specifically, word clouds, vader sentiment, and topic modeling was determined for each twitter screen name. Finally, word clouds and sentiment measures were repeated on the overall dataset.

## Stop Words

Two set of stop words[[12]](#footnote-12) were utilized during exploration, and later analysis:

* stopwords: general stop words for topic modeling and vectorization
* stopwords\_topics: combined with general stopwords and for topic modeling

## Topic Model

Topic Modeling (TL)[[13]](#footnote-13) was implemented against the five financial analysts mentioned above. Specifically, the corresponding twitter accounts were fed into the twitter API[[14]](#footnote-14) using the Twython[[15]](#footnote-15) python package. Then, the collected data was tokenized[[16]](#footnote-16) using CountVectorizer[[17]](#footnote-17) to obtain the term frequency (TF):

 **(equation 1)**

Specifically, the TF is the ratio of word occurrences divided by the total number of terms in the given document, inputted into the LatentDirichletAllocation[[18]](#footnote-18) method.

### Latent Dirichlet Allocation

The implemented codebase[[19]](#footnote-19) provides the ability to utilize a deterministic Non-Negative Matrix Factorization (NMF), and the probabilistic Latent Dirichlet Allocation (LDA) for topic modeling. While a comparison of the approaches could be analyzed, the benefits would not significantly outweigh additional computing. Without loss of generality, only the LDA was used. Furthermore, since latent variables are inferred (rather than observed) through iteration and maximization steps, the overall model follows a generative pattern.

 **(equation 2)**

However, to better understand the generative process, consider a simple case – predicting the topic of a token (x). This can be expressed as the joint probability of the word and topic:

 **(equation 3)**

In the above (equation 3), θ represents the per document topic distribution, and β the per corpus topic distribution. Furthermore, the likelihood component denotes the distribution of words for a given topic, while the prior signifies the number of topics for a given document. Moreover, the product of all token probabilities represents the probability a given document belongs to a specific topic[[20]](#footnote-20). LDA can be generalized and expressed as a distribution:

 **(equation 4)**

However, both θ and z are unknown hyperparameters, so approaches such as maximum likelihood estimation (MLE) cannot be performed directly. Instead, the Expectation-Maximization (EM) LDA[[21]](#footnote-21) was utilized. This process involves iteratively computing latent variables of the posterior distribution from equation 4. Since derivation of the posterior cannot be computed easily, an alternative posterior is used in place:

 **(equation 5)**

In this step, inferential statistic is used to approximate the best γ, and φ, minimizing the difference with the true posterior distribution[[22]](#footnote-22).

### Selected Topics

While the number of topics was chosen to be 10 for each financial analyst, an elbow method could be implemented to dynamically determine an appropriate number of topics. However, due to limited compute resources and timeline, this is left for future enhancements. Results from the below exploratory step, determined the associated quandl[[23]](#footnote-23) code for later analysis:

* FNYX\_QQQ
* FNSQ\_SPY
* BATS\_AMZN
* BATS\_GOOGL
* BATS\_AAPL
* BATS\_NFLX
* BATS\_MMT
* FNYX\_MMM
* PET\_RWTC\_D
* PR\_CON\_15YFIXED\_IR
* PR\_CON\_30YFIXED\_APR

Furthermore, for completeness additional codes were later analyzed:

* CBOE\_VX1
* COMP-NASDAQ

|  |  |
| --- | --- |
|  |  |
| Figure 6: LDA for JimCramer. | **Figure 7:** LDA for JimCramer. |

|  |  |
| --- | --- |
|  |  |
| Figure 2: word cloud for LizAnnSonders. | **Figure 3:** word cloud for LizAnnSonders. |

|  |  |
| --- | --- |
|  |  |
| Figure 4: word cloud for ReformedBroker. | **Figure 5:** word cloud for ReformedBroker. |

|  |  |
| --- | --- |
|  |  |
| Figure 8: LDA for SJosephBurns. | **Figure 9:** LDA for SJosephBurns. |

|  |  |
| --- | --- |
|  |  |
| Figure 10: LDA for TheStalwart. | **Figure 11:** LDA for TheStalwart. |

## Sentiment Analysis

Tweets associated with financial analysts were measured for positive, negative, and neutral sentiments. Using the python Vader package[[24]](#footnote-24), tweets were found more neutral, then positive, and negative, respectively. Next, aggregated words by topic from LDA, were used for sentiment measures for each financial analyst[[25]](#footnote-25). Since 10 topic models were generated, each x-label corresponds to a computed topic, with three associated measures of sentiment varying along the y-axis. Like the twitter sentiments, the overall topic models have similar Vader sentiment measures – neutral, positive, and negative.

|  |  |
| --- | --- |
|  |  |
| Figure 12: twitter sentiment for JimCramer. | **Figure 13:** lda sentiment for JimCramer. |

|  |  |
| --- | --- |
|  |  |
| Figure 14: twitter sentiment for LizAnnSonders. | **Figure 15:** lda sentiment for LizAnnSonders. |

|  |  |
| --- | --- |
|  |  |
| Figure 16: twitter sentiment for ReformedBroker. | **Figure 17:** lda sentiment for ReformedBroker. |

|  |  |
| --- | --- |
|  |  |
| Figure 17: twitter sentiment for SJosephBurns. | **Figure 18:** lda sentiment for SJosephBurns. |

|  |  |
| --- | --- |
|  |  |
| Figure 17: twitter sentiment for TheStalwart. | **Figure 18:** lda sentiment for TheStalwart. |

# Analysis

A baseline exploration was conducted on the Nasdaq and VIX index, then expanded to other indices to determine – “Can market sentiment from financial analysts predict stock prices”? To begin, the upper threshold limit (l) is determined for each financial analyst (A), for a given stock index or volume (xi):

 **(equation 6)**

Once the upper limit (l) is determined, the associated threshold(s) are computed:

 **(equation 7)**

Finally, a moving average is taken with several considerations. First, a lag parameter (L) is used to smooth the data, possibly eliminating trend. This value is alternatively known as a moving average (MA) or sliding window. To begin, the filter average (favg) and filter standard deviation (fstd) are computed for each threshold (T) on a given financial analyst (A):

 **(equation 8)**

 **(equation 9)**

A threshold parameter (T) defines the number of standard deviations above MA, which a point (Ai) is classified as a signal (sj)[[26]](#footnote-26).

 **(equation 10)**

 **(equation 11)**

Using equation 10, when Bj > Cj, the corresponding average and standard deviation filters from equation 8 and equation 9 are updated using the influence parameter (I):

**(equation 12)**

The influence parameter provide context whether past signals exceeding a z-score threshold, should be used to determine successive threshold values. Values range between [0, 1], where zero values have no influence on calculating successive threshold values, while a value of one is weighted with greater weight. Conversely, when Bj <= Cj, equation 12 adjusts by incorporating the current value:

**(equation 13)**

By default, if no threshold is computed, a default behavior is applied. Specifically, stock prices or volume measures (xi) were binned into classes when individual values exceed a previous defined step, namely (up, down) 🡪 (0, 1). Thus, “up” was assigned if the current day Index Value was greater than the previous day; otherwise, assigned down:

 **(equation 14)**

Next, tweets from financial analysts (X) were vectorized using term frequency-inverse document frequency (TFIDF). Data was split into 80% train, and 20% test. Bernoulli Naïve Bayes (BNB), Multinomial Naïve Bayes (MNB), and Support Vector Machine (SVM) were trained against the earlier binned classes.

Additionally, using earlier determined Vader timeseries data methodologies, including Granger causality test, ARIMA, and Long Short-Term Memory (LSTM) were applied for timeseries analysis on sentiment. While the Granger test is ideal for the ARIMA model, autocorrelation (ACF) and partial autocorrelation (PACF) were not utilized to determine appropriate (p,q,d) components for stationarity. Moreover, a future study may utilize both ACF and PACF to find optimal hyperparameters. However, a custom grid search method could provide an automated method that could implicitly resolve stationarity.

# Baseline Results

## Time series

## Granger Causality

## Classification

# Conclusions

1. <https://www.investopedia.com/terms/v/volatility.asp> [↑](#footnote-ref-1)
2. <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets> [↑](#footnote-ref-2)
3. <https://developer.twitter.com/en/apps> [↑](#footnote-ref-3)
4. <https://twython.readthedocs.io/en/latest/> [↑](#footnote-ref-4)
5. <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html> [↑](#footnote-ref-5)
6. <https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline.html> [↑](#footnote-ref-6)
7. <https://github.com/jeff1evesque/ist-736-hw/tree/master/data> [↑](#footnote-ref-7)
8. <https://developer.twitter.com/en/docs/basics/rate-limiting.html> [↑](#footnote-ref-8)
9. <https://www.quandl.com/tools/python> [↑](#footnote-ref-9)
10. <https://github.com/jeff1evesque/ist-736/blob/9652d7aa79dc576ca5ad671effbb76362beaa72a/app.py#L227> [↑](#footnote-ref-10)
11. <https://en.wikipedia.org/wiki/Domain_of_a_function> [↑](#footnote-ref-11)
12. <https://github.com/jeff1evesque/ist-736/blob/master/utility/stopwords.py> [↑](#footnote-ref-12)
13. <https://github.com/jeff1evesque/ist-736/blob/master/resources/topic-modelling-with-scikitlearn.pdf> [↑](#footnote-ref-13)
14. <https://developer.twitter.com/en/docs/tweets/search/overview/standard.html> [↑](#footnote-ref-14)
15. <https://twython.readthedocs.io/en/latest/> [↑](#footnote-ref-15)
16. <https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/topic_model.py#L110-L117> [↑](#footnote-ref-16)
17. <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html> [↑](#footnote-ref-17)
18. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html> [↑](#footnote-ref-18)
19. <https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/topic_model.py> [↑](#footnote-ref-19)
20. <https://github.com/jeff1evesque/ist-736/blob/master/resources/nlp_lecture_12-04-13.pdf> [↑](#footnote-ref-20)
21. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html> [↑](#footnote-ref-21)
22. <https://github.com/jeff1evesque/ist-736/blob/master/resources/research_exam09.pdf> [↑](#footnote-ref-22)
23. <https://www.quandl.com/> [↑](#footnote-ref-23)
24. <https://pypi.org/project/vaderSentiment/> [↑](#footnote-ref-24)
25. <https://github.com/jeff1evesque/ist-736/blob/master/brain/controller/topic_model.py> [↑](#footnote-ref-25)
26. <https://stackoverflow.com/a/22640362> [↑](#footnote-ref-26)