**Sentiment Analysis Model Comparison: The Case of Tesla**

Natural Language Processing for the Social Sciences (GR5067)

Group A

Belen Gutierrez, Binpeng Xiu, Marc Castillo, Siqi Zhao, Zehui Wang, Or’el Anbar

Columbia University

**Abstract**

This paper aims to evaluate the usefulness of social media sentiment as a predictor for stock returns. The research centers its analysis on public company Tesla, one of retail investors favorite stocks[[1]](#footnote-0), and whose CEO Elon Musk is active on social media platforms. Both these facts create an interesting and extensive pool of data points that the paper exploits to extract sentiment analysis and further study its correlation with stock market performance.

Data aggregation combines (1) YTD Tweets (from January - October 2020) containing the word ‘Tesla’. These were extracted by crawling the web for the tweet URLS (snscrape call in Python), which granted us access to over 340,000 tweets. (2) YTD Tesla EOD (End of Day) stock prices as well as NASDAQ closing prices from Yahoo Finance.

The paper concludes Twitter stock sentiment on Tesla marginally improves the performance of stock return prediction. This judgment is made after contrasting different sentiment analysis libraries and stock prediction models, deploying various parameters to minimize error rates by contrasting results.

**Introduction**

The focus of the project is to leverage the data preprocessing and modeling techniques learnt during the 2020 Fall semester in QMSS 5067 NLP to research whether stock sentiment from Twitter is an informative parameter in predicting Tesla stock performance.

The project consists of 3 phases: (1) Preprocess training data and training a sentiment analysis extraction model, (2) Sentiment Extraction from our Tesla Stock dataset, and (3) Stock return prediction.

1. After contrasting different possible training data sets, Sentiment140, consisting of over 1.6 million tweets from 2009 was deemed adequate to train our model given its size, and format; It presents the same data preprocessing challenges as our test data such as special characters. The data was transformed with NLTK tokenizers and stemmers. TF-IDF later created a vector matrix that set the grounds to perform our research. We then trained a Naïve Bayes classifier to generate stock sentiment on this dataset, which we subdivided earlier on between training and testing. Accuracy score obtained was around 80% for both Positive and Negative classes.
2. Sentiment extraction from our Tesla tweets:

First we extracted stock sentiment with the trained model from Phase 1 on our own custom dataset of tweets containing “Tesla.” We then did the same using VADER in order to compare which sentiment signals were more informative of Tesla’s stock return.

1. Stock return prediction was motivated by not only the team member’s will to apply the material covered in class but also our interests in finance, financial modeling and machine learning techniques for estimation. The models applied are Linear Regression and Random Forest.

**Data Description**

**Source 1: Sentiment140**

In order to train our own classifier for sentiment analysis as part of Phase 1, we needed a sufficiently large dataset of tweets that were already pre-labeled with sentiment scores. Otherwise, we would have no source of truth with which to evaluate model performance via traditional classification metrics.

Our solution was Sentiment140[[2]](#footnote-1), which is a publicly available dataset of tweets created by Computer Science graduate students at Stanford University. It consists of 1.6 million tweets that span the timeframe of April to June 2009. The dataset builders used a technique called distant supervision (Go et al. 2009) to construct the sentiment labels such that negative tweets were labeled as 0 and positive tweets were labeled 1. The website from which Sentiment140 can be downloaded also has an additional set of 498 manually labeled test set tweets, which provided an extra source of model evaluation that made the dataset nearly ready for our classification task.

**Source 2: Tesla Tweets**

After sufficient classification performance on the Sentiment140 dataset, we were looking to predict tweet sentiment in our own, custom-made dataset of tweets containing #Tesla. The remainder of this data description section details how the dataset was constructed before subsequent sections dive deeper into the actual sentiment analysis.

Twitter’s official API[[3]](#footnote-2) has traditionally been used for tweet-level research, and its search endpoint is how datasets like Sentiment140 were in the past. However, due to changes that occurred alongside Twitter’s API v2 that rolled out this past October, users with free developer accounts were no longer able to search for tweets that satisfied a particular query farther back than a week before the search request. Various third-party API wrappers were also rendered obsolete.

To circumvent this constraint so that we could fetch Tesla-related tweets from the past year, we utilized a command-line tool called snscrape[[4]](#footnote-3) that could scrape for tweet URLs given a query and save results as a text file. Then, free developer accounts could access a different official API endpoint to access the actual tweet information (namely the timestamp of posting and the actual tweet text) given the tweet ID found in the URL. We utilized the Python library Tweepy, which made API calls easier to write.

Using this web scraping method, we were able to collect tweets containing “Tesla” that were posted within January 1st to October 31st 2020. We then applied the preprocessing steps described before and dropped tweets with missing or incomprehensible (as well as non-English) text. The resulting sample size was 345,605 tweets. After sentiment extraction at the tweet level, the final data wrangling step was to aggregate tweet sentiment into a daily measure so that this dataset could be merged with Tesla and NASDAQ[[5]](#footnote-4) closing prices for the stock return prediction phase of our project. Our final date-level dataset spanned 194 trading days over the past year.

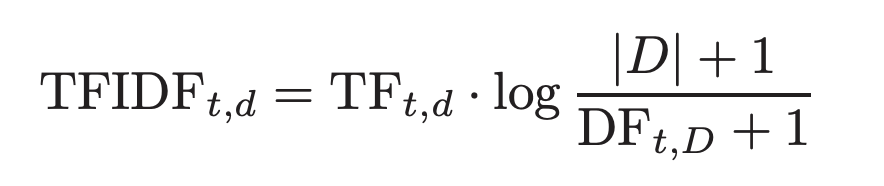
**Methodology**

**Preprocessing Steps**

First, however, we had to apply preprocessing steps. We made a custom function to strip tweet texts of their URLs and any user Twitter handles, replacing them with whitespace so that they wouldn’t contribute as sentiment signals. We also converted tweets to lowercase because, with the length of any given tweet being relatively short, we didn’t think our model should be treating words that begin sentences as meaningfully different from those same words placed elsewhere.

We then applied the NLTK library’s implementation of the common Porter algorithm for suffix stripping Porter (1980) to treat words with a common stem as the same feature. While stemming in this way is generally regarded as an inferior approach to the “smarter” lemmatization, which takes into account the morphology of a word (i.e. its context in a sentence, such as its part of speech), studies have shown marginal performance differences in information retrieval between the two preprocessing steps (Balakrishnan 2014). So, we preferred Porter stemming because of its less computationally expensive approach, which was valuable considering the size of our training set.

Next was vectorizing our text for feeding into machine learning models downstream, for which we used the staple Term Frequency—Inverse Document Frequency (TFIDF) vector space model. It considers two documents as similar if they share rare, but informative words, with its formula shown below:



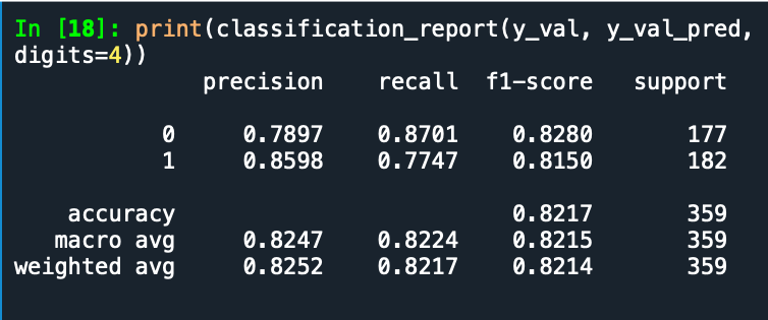
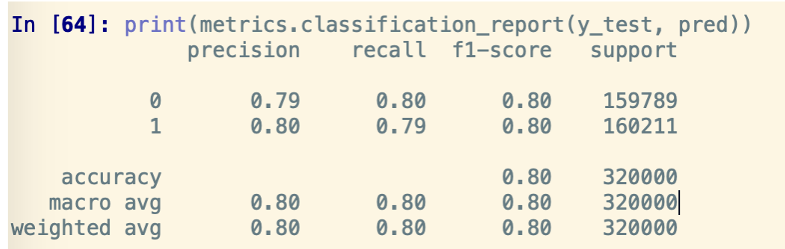
TF(*t*, *d*) represents the total number of occurrences of term *t* in document *d* and DF(*t*, *d*) represents the total number of documents in which term t occurs. This weighting scheme scales down the importance of very common terms, which makes this model a sensible choice for our eventual goal of extracting the sentiment of a dataset of Tesla tweets for stock price prediction. We further tweaked the scikit-learn library’s out-of-box TF-IDF vectorizer by considering both unigrams and bigrams rather than exclusively unigrams, as for the purposes of sentiment extraction, accounting for bigrams would capture the distinction between, say, “cool,” (likely a positive sentiment) and “not cool” (likely negative). One tradeoff of this choice, however, is that doing so increases our feature space (Shahmirzadi et al. 2019). After fitting our vectorizer on the training set file, we ended up with a feature space of 298,024.

**Classifier Training and Evaluation**

We finally arrive at model training. We picked a Naïve Bayes (NB) classifier. While support vector machines (SVM) are widely used as a sentiment learning approach, NB has the advantage of training more quickly over large datasets and being more robust to noise that makes them less prone to overfitting, all while being sufficiently good for most document-level analysis (Narayanan et al. 2013). The model achieves these advantages via its simplifying conditional probability assumptions. That is, given a class (positive or negative sentiment), words are assumed to be conditionally independent of each other. With this prior assumption, NB models can output the class with the maximum posterior probability in linear time using maximum likelihood parameter estimation (Hasan et al. 2018).

After splitting the training data file into an 80% (1,280,000 tweets) sample for 5-fold cross validated training, we then measured NB classifier performance on both the 20% hold out set and on the separate test set file of manually labeled tweets. Screenshots of our evaluation results are shown below:

**Figure 1 : Naive Bayes Model Evaluation Screenshots**



*On the left: hold out set accuracy and averaged F1-scores hovered around 80%. On the right: test set file’s accuracy and averaged F1-scores hovered around 82%*

Interestingly, the classifier performed better on the manually labeled test set file. One caveat, however, is that we had to drop a few of the observations there that were labeled “neutral” because our NB classifier is binary.

**Sentiment Extraction**

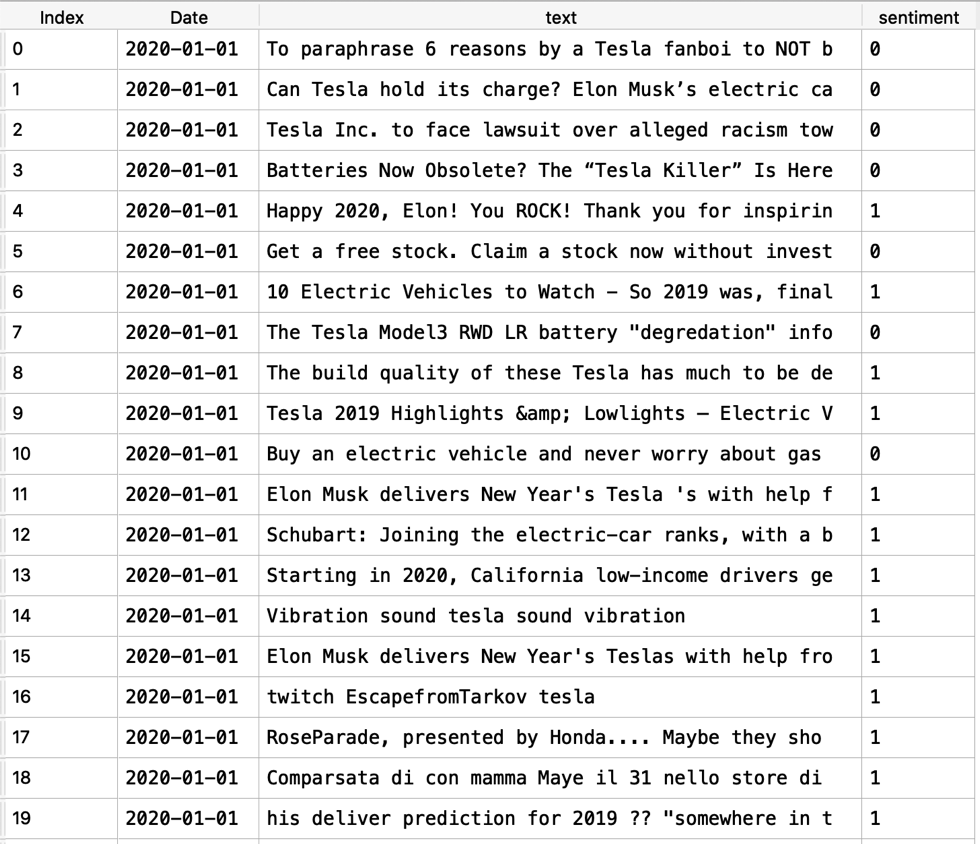
We employed two approaches to perform the sentiment analysis and assign sentiment scores to our Tesla-related tweets dataset.

The first approach was based on the Naïve Bayes Classifier as described above, which was trained on the Sentiment140 dataset. The second approach was to use The Valence Aware Dictionary and Sentiment Reasoner (VADER) in Natural Language Toolkit (NLTK) using Python[[6]](#footnote-5).

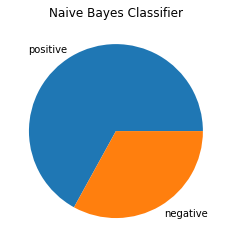
**Naïve Bayes Classifier**

We applied the Naïve Bayes classifier trained as above to the Tesla-related tweets dataset. The Naïve Bayes model classified tweets into two categories: positive (1) and negative (0). Table 1 shows a snippet of the sentiment classification results and Figure 2 shows the allocation of the tweets of the two polar sentiments. We then calculated the percentage of negative tweets for each day. This information can be used as a predictor for the Tesla stock return prediction modeling in the next section.

**Table 1 : Polar Sentiment Results Using Naïve Bayes Classifier**



**Figure 2 : Sentiment Allocation Using Naïve Bayes Classifier**

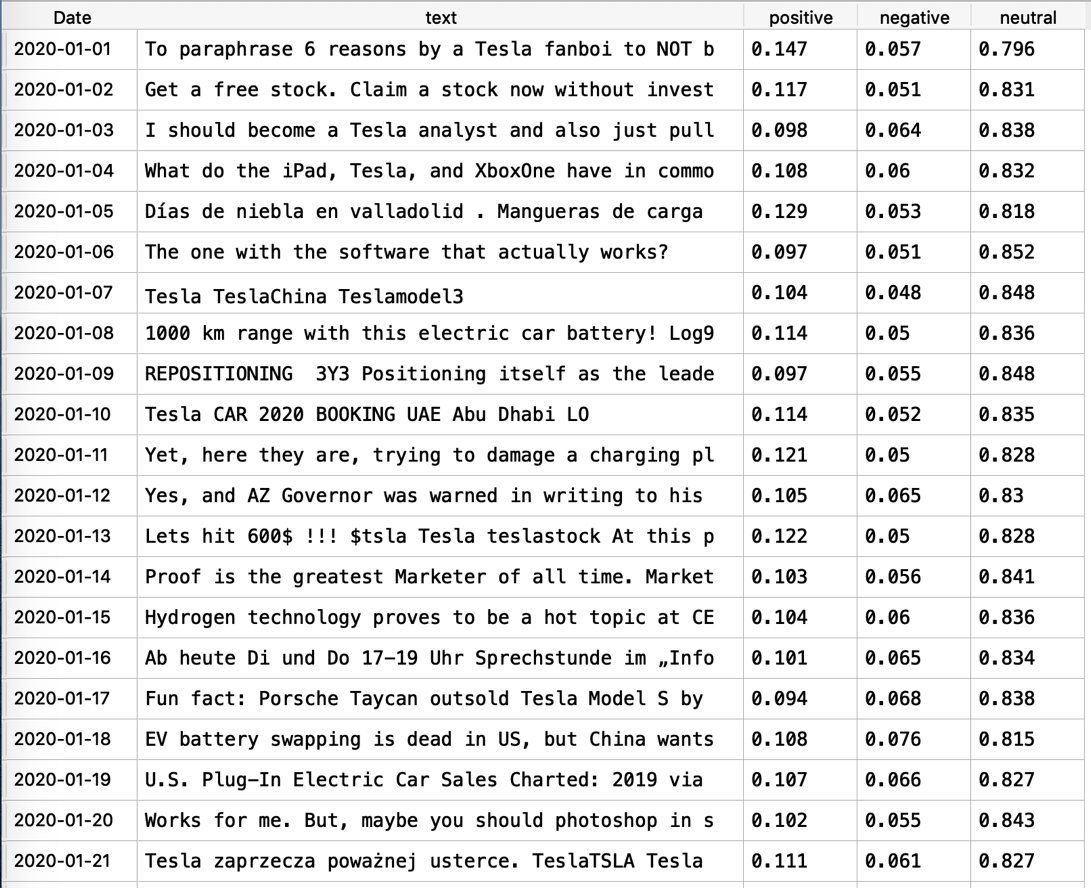


**VADER**

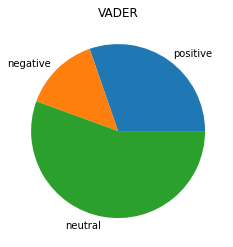
VADER, unlike the Naïve Bayes Classifier, is a lexicon-based approach, which does not require training. The VADER sentiment scores are calculated based on the pre-prepared lexicon of sentiment-related words. Using its specially designed VADER model, we can calculate the sentiment scores for the concatenated tweets of each day. It is also worth mentioning that VADER is uniquely tuned to accommodate language commonly used on social media. It takes into consideration emoticons, slang, intensifiers, acronyms, etc., in addition to commonly used sentiment-related words (Hutto and Gilbert 2014).

We applied VADER to the Tesla-related tweets dataset and got the sentiment scores for the tweets of each day. This exercise yielded three types of outputs: positive, negative, and neutral scores. We did not use other VADER output elements such as the compound score. Since many tweets had the same compound score, we decided to omit this measure from our analysis since it would not be helpful in the modeling process later. Table 2 shows the polar sentiment scores using VADER, and Figure 3 shows allocation of sentiments in the Tesla tweets.

**Table 2 : Polar Sentiment Results Using VADER**



**Figure 3 : Sentiment Allocation Result Using VADER**



**Tesla Stock Return Prediction Model**

After getting the sentimental scores, we wanted to test whether they can help to improve the model to predict the TESLA stock return. We used four feature sets in this step. The four feature sets are:

1. Base. Use last day volume and last day Nasdaq index return.
2. Vader. Add compound score to Base.
3. Naïve Bayes. Add positive, negative, neutral scores to Base.
4. All. All Naïve Bayesian, VADER, Base features together.

**Model Evaluation Criteria**

In this project, we chose to use Root Mean Square Error (RMSE) as the criteria to evaluate the prediction. Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of the distance between the regression line and the data point; RMSE is a measure of the spread of residuals. In other words, RMSE measures how concentrated the data is around the line of best fit.

A picture containing text, clock

Description automatically generated

**Linear Ridge Regression Model**

We split the whole data set into 144 train samples and 48 test samples. For the linear model, we select the Ridge Regression Model. A Ridge Regression is a regression that mitigates the problem of multicollinearity in a linear regression, which commonly occurs in models with large numbers of parameters. The way to calculate the coefficient of Ridge Regression is given below.



To select the best alpha in the Ridge Regression Model, we used rolling validation to find the alpha which maximizes the RMSE in the validation set to be as alpha.

Rolling validation can be illustrated in the figure below:

**Figure 4 : Rolling Validation Graphical Representation**

Table

Description automatically generated

For the time-series data, we used rolling validation rather than cross-validation, to avoid looking-ahead bias. In the graph above, we can see that we select a time window to train, and use another time window as validation. After this sample, we moved to the starting point of the training set forward and generated another sample. This is rolling validation. For the ridge data, we find the graph for alpha is as below:

**Figure 5 : Linear Ridge Regression RMSE**

Chart, line chart

Description automatically generated

Given the results illustrated above, we determined the best alpha to achieve a stable lowest RMSE is at 2.3357. We used this alpha and the training model at training set before testing the results with the test set Following this exercise, we got the RMSE of four feature sets as below:

**Figure 6 : RMSE Comparison of Different Feature Sets - Linear Ridge Regression**

Chart, bar chart

Description automatically generated

We can find that VADER achieves a lower RMSE than the Base. The compound score using the VADER method does help to improve stock return prediction for TESLA. Naïve Bayes and All get higher RMSEs than Base, so the positive, negative, neutral scores using the Naïve Bayesian method don’t help to improve stock return prediction for TESLA.

**Random Forest Model**

We also applied a Random Forest Regressor Model to examine whether using Twitter Sentiment Analysis is helpful for the prediction of Tesla stock return. A Random Forest model fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control overfitting. For the Random Forest Regressor, we decided to tune two parameters, which are “*max\_depth*” and “*n\_estimators*”. We tune the “*max\_depth*” because the “*max\_depth”* of a tree in Random Forest is defined as the longest path between the root node and the leaf node. Generally, as the max depth of the decision tree increases, the performance of the model over the training set increases continuously. Alternatively, as the “*max\_depth”* value increases, the performance over the test set increases initially but after a certain point, it starts to decrease rapidly. From the graph below, we are able to see that “*max\_depth*” should be 90 since we want to make the RMSE as small as possible.

**Figure 7 : Random Forest RMSE against Tree Depth**

Chart, line chart

Description automatically generated

The second variable we tuned was the “*n\_estimators*.” The “n\_estimators” is the number of trees we would build. Generally, more trees yields better performance. From the graph below, we are able to see that “*n\_estimators*” should be around 115 since we want to make the RMSE as small as possible.

**Figure 8 : Random Forest RMSE against Number of Estimators**

Chart, line chart

Description automatically generated

In the next step, we used the tuned parameters to train our model at the training dataset, then we tested the results using the testing dataset. We got the RMSE results of our four featured sets, illustrated below:

**Figure 9 : RMSE Comparison of Different Feature Sets - Random Forest**

Chart, bar chart

Description automatically generated

By observing the RMSE results, we can see that:

1. VADER achieves a lower RMSE than Base, so the compound score using the VADER method does help to improve stock return prediction for TESLA.
2. Naïve Bayes and All get higher RMSEs than Base, so the positive, negative, neutral scores using the Naïve Bayesian method don’t help to improve stock return prediction for TESLA.

**Model Comparison**

From the analysis above, we can see that both Ridge Regression and Random Forest Regression have comparatively low RMSEs. Comparatively, Ridge Regression has lower RMSE than the Random Forest Regression.

**Results**

From Phase 2, we found that the Naïve Bayes Model achieved 80% accuracy in the five-fold cross validation and 82% accuracy in the testing set. Also it had around 80% in both precision and recall. We inferred that the trained Naïve Bayes classifier can result in a pretty stable classification to use in the Tesla-related tweets dataset.

In Phase 3, after tuning the models, we found that adding the VADER sentimental score helps to achieve a lower RMSE in both Ridge regression and Random Forest. Considering this finding, we can determine that the compound score using the VADER method does help to improve stock return prediction for TESLA. On the other hand, the sentiment scores from Naïve Bayesian and All get higher RMSEs than the Base case. This being the case, the positive, negative, neutral scores using the Naïve Bayesian method don’t help to improve stock return prediction for TESLA. Besides, Ridge Regression has a better performance because it has lower RMSE than the Random Forest model.

**Table 3 : RMSE results of Ridge and Random Forest**



**Conclusion**

The research question of our project is to investigate if the sentiment scores help better predict the stock return model. This project focuses on the Tesla company.

From all the analysis above, we saw improved performance when including the VADER polar sentiment scores but not with our own customized classifier. Since the Naïve Bayes model is an oversimplified model trained on a relatively small size of training data, it is not surprising that our classifier does not help in the prediction. With the improved performance of including VADER sentiment scores, we can conclude that carefully crafted sentiment scores do improve the stock return prediction model.

Retrospectively, we saw potential for improvement in our customized Naïve Bayes classifier. Currently, our classifier only contains two classes. In future research, we would consider other classifiers on a bigger training dataset. We would also consider other preprocessing choices including changing the TF-IDF properties.

**References**

Balakrishnan, Vimala & Ethel, Lloyd-Yemoh. (2014). Stemming and Lemmatization: A

Comparison of Retrieval Performances. Lecture Notes on Software Engineering.

2. 262-267. 10.7763/LNSE.2014.V2.134.

Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant

supervision. CS224N project report, Stanford, 1(12), 2009.

Hasan, A., Moin, S., Karim, A., & Shamshirband, S. (2018). Machine Learning-Based

Sentiment Analysis for Twitter Accounts. Mathematical and Computational

Applications, 23(1), 11. doi:10.3390/mca23010011

Hutto, C., & Gilbert, E. (2014). “VADER: A Parsimonious Rule-based Model for

Sentiment Analysis of Social Media Text,” In International AAAI Conference on Web and Social Media. Retrieved from https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109/8122

Narayanan, V., Arora, I., & Bhatia, A. (2013). Fast and Accurate Sentiment

Classification Using an Enhanced Naive Bayes ModelLecture Notes in Computer Science, 194–201.

Porter, M.F. (1980), "An algorithm for suffix stripping", Program: electronic library and

information systems, Vol. 14 No. 3, pp. 130-137. https://doi.org/10.1108/eb046814

Shahmirzadi, O., A. Lugowski and K. Younge, "Text Similarity in Vector Space Models:

A Comparative Study," 2019 18th IEEE International Conference On Machine

Learning And Applications (ICMLA), Boca Raton, FL, USA, 2019, pp. 659-666,

doi: 10.1109/ICMLA.2019.00120.

1. https://www.investopedia.com/retail-faves-are-beating-hedge-fund-mutual-fund-picks-5024920 [↑](#footnote-ref-0)
2. http://help.sentiment140.com/for-students [↑](#footnote-ref-1)
3. https://developer.twitter.com/en/docs/twitter-api [↑](#footnote-ref-2)
4. https://github.com/JustAnotherArchivist/snscrape [↑](#footnote-ref-3)
5. finance.yahoo.com [↑](#footnote-ref-4)
6. https://www.nltk.org/index.html [↑](#footnote-ref-5)