Twitter Sentiment Analysis: A Learning Approach

**Gabe Smith**

Department of Physical and Mathematical Sciences

Brigham Young University

Provo, UT

[gabessmith432@gmail.com](mailto:gabessmith432@gmail.com)

Table of Contents

[Abstract 1](#_Toc38456255)

[Existing Research 2](#_Toc38456256)

[Approach/Method 3](#_Toc38456257)

[Data Set Information 3](#_Toc38456258)

[Data Cleaning Methods 3](#_Toc38456259)

[Text Vectorization 4](#_Toc38456260)

[Results 5](#_Toc38456261)

[Random Forest Classification 5](#_Toc38456262)

[XGBoosting Classification 5](#_Toc38456263)

[Logistic Regression 6](#_Toc38456264)

[Multinomial Naïve Bayes Classification 6](#_Toc38456265)

[Overall Results 7](#_Toc38456266)

[Subsequent Development 7](#_Toc38456267)

[Conclusion 7](#_Toc38456268)

[References 8](#_Toc38456269)

# Abstract

Sentiment analysis is an extremely useful tool for finding overall opinion regarding a specific topic. Various organizations and industries use sentiment analysis models and methodology to understand sentiment concerning their organization or products. Its applications are almost endless; whether it be government agencies trying to understand public sentiment before their elections or businesses understanding overall opinion of their product or company, organizations need to understand sentiment regarding themselves. As social media popularity and presence continues to grow, sentiment analysis will become more powerful and useful in giving accurate sentiment polarity feedback.

This paper attempts to conduct sentiment analysis on labeled Twitter data using vectorization and machine learning. The report first addresses existing approaches and prior methods, explaining where this approach fits in with current research. Data cleaning technicality following tweet vectorization are explained to show how the data is prepared for machine learning. Then various machine learning algorithms are applied to this training set and results are then analyzed. Next steps, conlusions regarding results and an annotated bibliography end the paper

# Existing Research

Sentiment analysis overall interest, approaches and models have grown with the rise of technology and the Internet. Very little sentiment analysis existed up until the late 1990’s as most texts were not uploaded onto computers at the time. However, sentiment analysis had its explosive start when the .com bubble burst in the early 2000’s, and since then more and more texts have become available on the Web [1]. Thus around 99% of papers have been published after this time period. Despite its recent growth and development, close to 7,000 papers exist on the subject from just the past ~15 years. Hence many approaches have already been tried and tested, however it is useful to analyze and understand these methods to potentially grow upon them.

One popular paper uses both a lexicon and machine learning-based approach to predict sentiment on both movie and software reviews [2]. A lexicon-based model looks at the ordering of words and small phrases to analyze word sentiment polarity rather than entire statements for a model. However, its lexicon analysis includes ~7,000 words which could be potentially small depending on the unique words and size of the data. However, it then plugs in results from the lexicon-based approach into the machine learning models and found decent results. The lexicon-based approach alone only achieved on average ~70% accuracy, whereas the combined lexicon and learning approach reached ~85% accuracy and learning alone reached ~85%. The paper only used Support Vector machines, thus this paper focused on using various machine learning methods should show higher or similar results as found in this paper.

A second approach uses fuzzy ontology domain trees to predict sentiment for specific words. Many words have easy sentiment polarity, whether being strongly positive or negative, but numerous other words aren’t easily determined as positive or negative without strong domain knowledge. These “fuzzy” words are analyzed using ontology domain trees split to decrease ambiguity for these fuzzy words [3]. Thus this helps give accurate sentiment contrariety for specific words, but do not analyze entire statements for overall polarity.

Several very recent approaches use deep learning Python libraries such as keras or pytorch to predict sentiment analysis that have significantly higher accuracy than most other models. Many approaches use Embedding() functions that change each statement into a unique vector then use either Convolution Neural Networks (CNN) [4] or Recurrent Neural Networks (RNN) [5] to train and predict the sentiment polarity, often achieving above 90% accuracy. However for this project, the focus will be on increasing accuracy to achieve about 90% accuracy using more popular machine learning methods such as Random Forest, Multinomail Naïve Bayes or Logistic Regression.

# Approach/Method

Data Set Information

The data used for this project comes from an online Kaggle competition, with training data containing 1.6 million tweets and test data containing 500 tweets, with each row carrying tweet information regarding time, user and sentiment [6]. This data is extremely popular, which has been proven and used by various other sentiment analysis research papers to analyze the accuracy of different algorithms. The data did come without emoji’s, thus some information will be lost to help the predict sentiment polarity, however there shouldn’t cause substantial bias in the analysis.

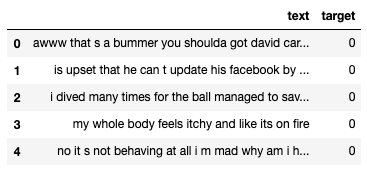
There are some differences between the two datasets: the training data only has labeled data for positive or negative tweets, whereas the test dataset contains sentiment labels for positive, neutral and negative tweets. For use of this project, the Python NLTK Vader sentiment analysis function was used to find whether neutral tweets contain greater positive or negative sentiment, then the tweets were then relabeled as such. However, several neutral tweets contained neither positive or negative sentiment, thus these were removed from the test dataset as the focus of this paper is to primarily increase model accuracy.

Data Cleaning Methods

In order to clean the dataset, significant data preprocessing was needed in order to take out all unneeded text, emoticons and other information in the data. Various methods of cleaning were found online due to the popularity of the dataset, thus best methods came after thorough analysis and selection for this approach [7]. All cleaning was accomplished using various libraries in Python including Regular Expressions, NumPy and Pandas. The cleaning was accomplished through 6 different steps:

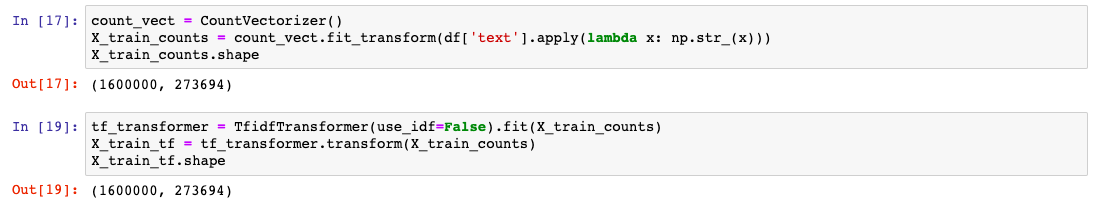
1. Removal of HTML Decoding: At times, tweets aren’t read correctly by pandas and HTML code is included in the tweets. However, this is a simple fix in Python using BeautifulSoup to take out these expressions.
2. Removal of ‘@’ mentions: All tweet mentions (‘@’ and user info) were removed due to the overwhelming majority having having overall neutral sentiment, thus not adding significant information to predicting positive or negative sentiment.
3. Removal of URL mentions: Similar to ‘@’ mentions, most URL mentions were almost equally positive and negative, thus without significant sentiment polarity add.
4. Removal of non-UTF-8 characters: Non-English characters existed in several tweets, thus these were promptly removed.
5. Removal of all punctuation: All punctuation including hashtags were removed for ease of analysis. However, hashtag phrases were kept as most contain significant sentiment information.
6. Convert words to lowercase: For ease of evaluation and to avoid duplication of capitol and lowercase lettered words, all words were converted to lowercase letters.

Code containing data cleaning functions is included in the Twitter\_Data\_Cleaning.ipynb file, however tweets were converted into this format for analysis:

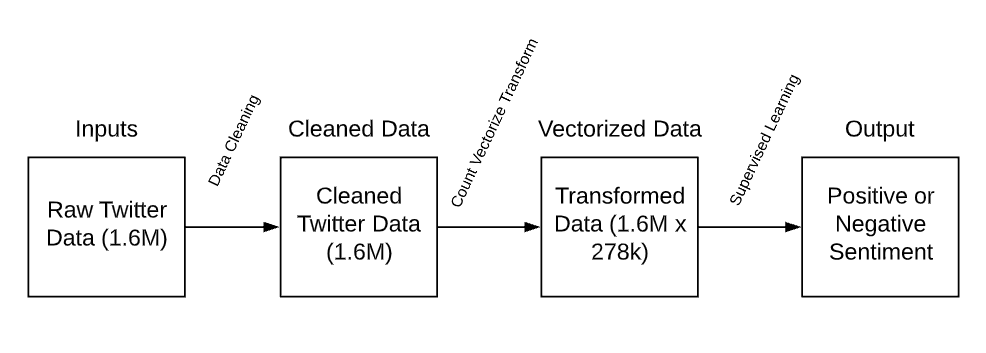


Text Vectorization

Following syntactical cleaning, text data was then transformed to represent word counts in each each tweet. The CountVectorize() and TfidfTransformer() functions from Python’s Scikit-learn library were used to enumerate the data in preparation for machine learning. This then transformed the matrix from one column of text into 200,000+ columns of small real values, where each column represented one specific word’s existence in each tweet. The transformation was very simple code, as outlined below:



Thus the dataset greatly increased in total unique values, therefore increasing model run-time but in hope of giving increasingly accurate results. The following flow chart diagrams the entire process for preparing, cleaning and recognizing semantic polarity:

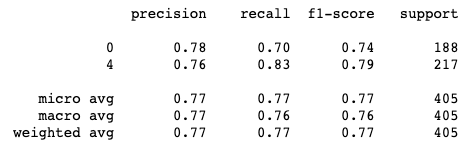


# Results

To help solve this problem, various machine learning methods were applied to the cleaned and vectorized data. Thus each model will contain binary classification metrics and results, then greater discussion of their meanings including strong points, shortcomings and overall accuracy.

Random Forest Classification

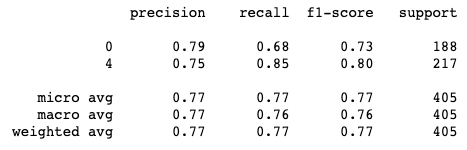
Random Forest Classification computes often hundreds of decision trees striving to label and understand splitting points in the data. Due to the size of the dataset and to the large numbers of random trees calculated and average to increase accuracy, Random Forest ran significantly slower. Tree based algorithms look create decision boundaries explaining differences between different leaf nodes, thus the 278,000 columns in the dataset forced trees to spend significantly longer time creating decision nodes. The results are shown below:



Upon first glance, it is interesting to note the higher accuracy in predicting positive (4) sentiment over negative (0) sentiment. The model seems to understand positive sentiment words or tweets significantly better than negative sentiment tweets, where the True Positive Rate or recall was 13% higher for positive vs negative tweets. However, the model didn’t perform as accurately as desired.

XGBoosting Classification

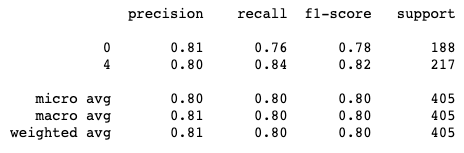
Similar to the Random Forest classification, the XGBoost model is a decision-based tree model that creates subsequent trees based off the misclassified datapoints of previous trees. This model is extremely powerful, often outperforming other tree-based models such as Random Forest classification, but has drawbacks in both significantly longer runtime and higher probability to overfit the data. As expected, the run time took significantly longer than the Random Forest classification, however the results were less than desired for this problem:



The model performed slightly better than Random Forest at classifying true positive sentiment but performed worse for predicting true negative. Thus we see similar results for each model, where XGBoost outperformed Random Forest in some areas but lacked the same in others, where both gave a similar overall accuracy of 77% for both models. This is no major surprise, as the model seems to struggle equally with the differing tree-based machine learning approach.

Logistic Regression

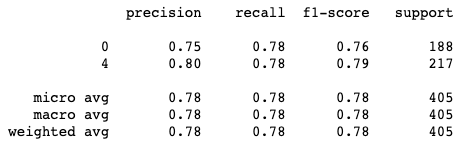
Logistic Regression is a powerful binary classification tool used to regress data using the logistic function. Thus this approach often runs faster than tree-based algorithms, often performing at the same leve or higher than tree-based models. As expected, the Logistic Regression algorithm ran 4x faster than the Random Forest Model, and delivered more accurate results:



Here the model still predicted true positives higher than true negatives, similar to previously discussed models, but overall predicted more accurately than the previous models. This may be due to the model finding coefficients more accurately for the differing words than decision trees randomly splitting columns entirely based on randomness. Thus regression-based techniques may have higher accuracy and faster runtime due to the high number of columns in the dataset.

Multinomial Naïve Bayes Classification

Multinomial Naïve Bayes classification uses Bayes theorem using the normal distribution to compute the probability of each point belonging to each target class and then assigns the data point to the class with the highest probability. Thus the model found quite different results compared to the previous models:



This model had very similar predictions for both true negatives and true positives, which are quite different than the previous classification models. However, the accuracy continues to be between 77-80% between all models thus far, where no specific model significantly outperforms the rest.

Overall Results

Thus based on these attempted models, the data seemed to hover around 77-80% accuracy based entirely on a learning approach. The Logistic Regression outperformed the other models by about 2-3% which isn’t a major jump considering the small size of the test dataset. Other models including Stochastic Gradient Descent were applied to this problem, but none reached above 80% accuracy computed by the Logistic Regression model. These results are found in the source code in the source file Twitter\_Dataset\_Vectorization\_and\_Model\_Results.ipynb.

# Subsequent Development

In conjunction with what was accomplished, the sub-average results show necessary future change of approach and models themselves to increase accuracy. The data was prepared using Vectorization and TFIDF transformations which create extremely large sparse matrices that have been historically difficult to deal with in machine learning [8]. Thus future approaches should either convert these sparse matrices to dense matrices or use Embedding methods to decrease the dimensions and sparcity of the data.

Another potential approach includes using deep learning methods to perform analysis as discussed under the Existing Research section. The models used for this problem were powerful yet deficient in many areas, proving themselves unable to find specific polarity for many of the tweets. Thus future approaches should focus on RNN or CNN models to recognize sentiment polarity.

# Conclusion

After looking deeply into the data, the model worked pretty well. It obtained similar results to the previous research using both lexicon and learning methods1, however the model results were very consistent across each algorithm, ranging between 77 – 80% accuracy. Thus no major developments, findings or changes have come from this approach, however the results prove the existing correlation between sentiment and existing word counts strictly word counts.

This conducted research by no means is ready for commercial use, as accuracy isn’t extremely high and many nuances exist in the computation. However, this paper rather serves as a springboard to potential areas of greater research and interest by the NLP community, where methods could be developed to further progress some of the findings in this paper. Hopefully others may use the findings in this paper to increase accuracy in recognizing sentiment to better their product or organization, to potentially better understand overall opinion.

# References

[1] Mika V. Mäntylä, Daniel Graziotin, Miikka Kuutila. “The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers,” [Online]. Available: <https://arxiv.org/pdf/1612.01556.pdf>. [Accessed: 21-Apr-2020].

[2] Andrius Mudinas, Dell Zhang, Mark Levene. “Combining Lexicon and Learning based Approaches for Concept-Level Sentiment Analysis,” [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/2346676.2346681>. [Accessed 30-Mar-2020].

[3] Lizhen Liu, Xinhui Nie, Hanshi Wang. “Toward a fuzzy domain sentiment ontology tree for sentiment analysis,” [Online]. Available: <https://ieeexplore.ieee.org/document/6469930>. [Accessed 30-Mar-2020].

[4] Jason Brownlee, “How to Get Started with Deep Learning for Natural Language Processing.” [Online]. Available: <https://machinelearningmastery.com/crash-course-deep-learning-natural-language-processing/>. [Accessed: 30 Mar, 2020].

[5] Artem Oppermann, “Sentiment Analysis with Deep Learning of Netflix Reviews.” [Online]. Available: <https://towardsdatascience.com/sentiment-analysis-with-deep-learning-62d4d0166ef6>. [Accessed: 30 Mar, 2020].

[6] “Twitter Sentiment Analysis.” [Online]. Available: <https://www.kaggle.com/c/twitter-sentiment-analysis2>. [Accessed: 10 Feb, 2020].

[7] Ricky Kim, “Another Twitter sentiment analysis with Python – Part 1.” [Online]. Available: <https://towardsdatascience.com/another-twitter-sentiment-analysis-bb5b01ebad90>. [Accessed: 30 Mar, 2020].

[8] Jason Brownlee, “A Gentle Introduction to Sparse Matrices for Machine Learning.” [Online]. Available: <https://machinelearningmastery.com/sparse-matrices-for-machine-learning/>. [Accessed: 30 Mar, 2020].