

Twitter Sentiment Prediction Model Use Cases

Introduction

The political conflict between Donald Trump, representing the Republican Party, and former Vice President Joe Biden, representing the Democratic Party was reflected in the discussions among users on online social networks like Twitter, Instagram, Facebook etc.

When you think of politicians and Twitter, chances are President Donald J. Trump comes to mind. Ever since he launched his run for office in 2015, Trump has become infamous for what many have described as derogatory, negative, and somewhat inflammatory tweets. Turns out that with even as little as 280 characters, he can communicate a whole spectrum of emotions, “facts”, and, believe it or not, opinions.

Over the past few years, the use of political Twitter accounts has skyrocketed. Nowadays it seems like every political leader and their entire family has a Twitter account, and they all feel the need to share their opinions with all of us on the platform. In fact, many leaders use Twitter as their primary mode of communication to the public rather than addresses or newsletters. However, this has caused some interesting problems. Often tweets aren’t vetted, and when coming from “personal” accounts rather than “official government-run communications” they can be quite polarizing and/or problematic.

Research Questions

We wanted to investigate how different leaders around the world used Twitter. Specifically, we oriented our analysis around the following questions:

1. How people are reacting to the daily news and relate that to presidential election?
2. What are the most frequently used keywords used in political tweets? How do these differ by candidate?
3. Can we predict the tweet sentiment using ML model and compare that with industry products like Google and TextBlob?

Twitter Sentiment Prediction Model Use Cases

Data



Our first step was to gather data. We knew we wanted to gather tweets that had key words “Trump” or “#Trump” or “Biden” or “#Biden” over 11 days at random time of a day. Due to infrastructure limitation, we took a sample of 10K tweets per day.

First, we tried Twitter’s native API but soon realized that they had quite strict limits on how much data a single person could download. We used twitter API to download the data.

```
▶ # Create API instance with authentication
api= tweepy.API(auth,wait_on_rate_limit=True)

▶ # Set keywords, hashtag or mentions tag and # of searches
keywords = '#Trump OR Trump OR Biden OR #Biden'
key_words =['Trump','Biden']

▶ # Function for make the search using Twitter API
def search_tweets(new_search):

    # performs the search using the defined variables
    for tweet in limit_handled(tweepy.Cursor(api.search,
                                             q=new_search,
                                             count=count,
                                             tweet_mode='extended',
                                             lang=lang,
                                             result_type=result_type,
                                             include_entities=include_entities).items(totalTweets)):

        try:

            # Checks if its a extended tweet (>140 characters)
            content = tweet.full_text
```

After downloading the data, we focused on cleaning up the tweets and translating them. We removed URLs, @ tags, punctuations regular expressions and normalized all of the tweets to be lower case. We used tweets that had 5 or more words.

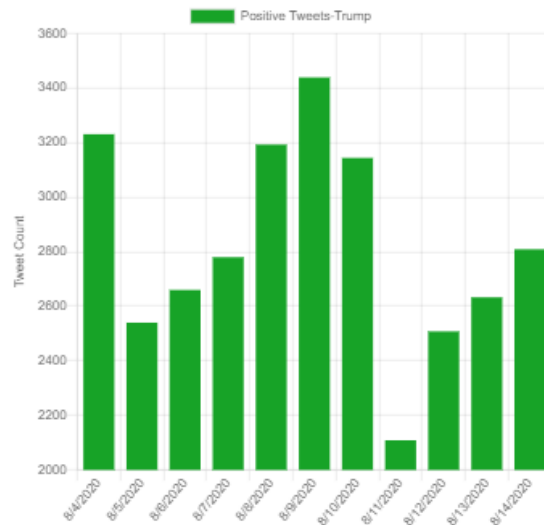
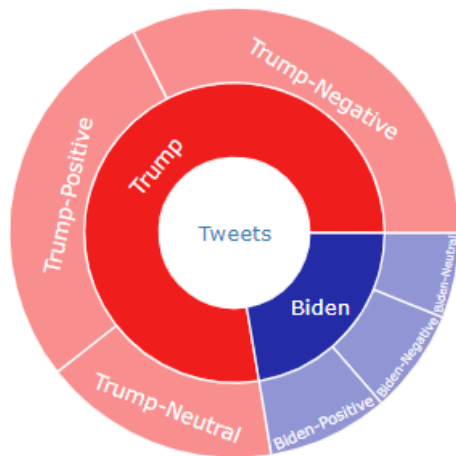
Twitter Sentiment Prediction Model Use Cases

Analysis

First, we wanted to analyze basic information. We were surprised to see tweets linked with President Donald Trump (77.64%) were significantly higher than Joe Biden (22.36%).

Trump vs Biden Tweet Classification (%)

Click on individual leaves to see date based tweet counts.



Tweets

- This is so sad. His mind is totally F.U. Thank you for viewing the trailer of Fractured Fairy Tales. Trump version.
<https://t.co/R0gmSO9dU4>
- 1 Why is Traitor Trump so eager to open schools?
- 2 @bawthetseal Donald Trump has blocked you* LMAO
- 3 #NATO #defense Trump, Germany and the Pom-American Grenadier <https://t.co/jlQJiVoUQf>
- 4 @richardmarx I need whatever the reporter is taking to keep his cool. What an exercise in patience. Trump is stammering like a 5 year old caught climbing a chair to reach cookies.
- 5

Next, we wanted to look at some of the keywords being used by the users. We limited this analysis to English accounts because, during translation, the text data can be changed enough to not accurately reflect word usage. Word clouds are a great way to easily visualize how certain keywords stand out in a large collection of text.

We used the WordCloud Python library to generate the images by feeding in the raw tweet data.

Twitter Sentiment Prediction Model Use Cases

```
# Word Cloud for President Trump - Positive
all_words = pd.Series([t for t in df_trump[df_trump.Sentiment == "Positive"].Tweet]).str.cat(sep=' ')
wordcloud = WordCloud(width=800, height=500, background_color='white',
                      random_state=420, max_font_size=110).generate(all_words)

plt.figure(figsize=(10, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

President Donald Trump



Former Vice President Joe Biden



After gathering our baseline data, we shifted our focus to sentiment analysis and decided to use VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. This provided us with four data points for each tweet – Positive, Negative, Neutral and Compound score.

Twitter Sentiment Prediction Model Use Cases

We compared positive and negative scores to determine final sentiment.

```
# import SentimentIntensityAnalyzer class from vaderSentiment.vaderSentiment module.
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# function to print sentiments
# of the sentence.
def getSentiment(sentence):

    # Create a SentimentIntensityAnalyzer object.
    sid_obj = SentimentIntensityAnalyzer()

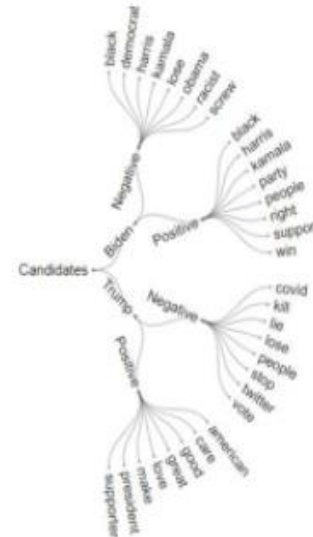
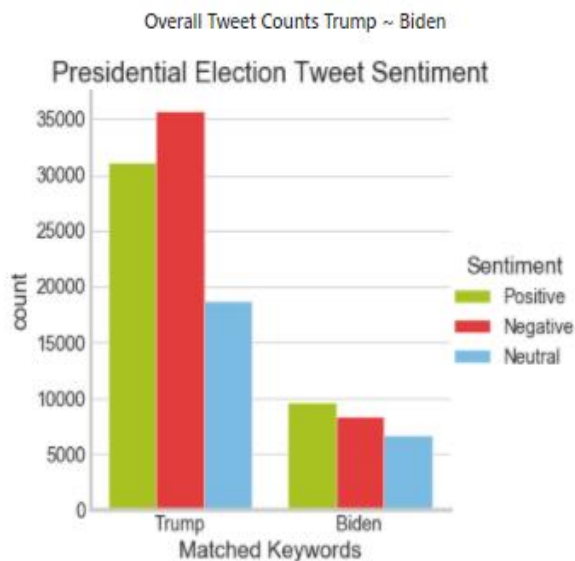
    # polarity_scores method of SentimentIntensityAnalyzer object gives a sentiment dictionary which contains
    sentiment_dict = sid_obj.polarity_scores(sentence)

    # decide sentiment as positive, negative and neutral
    if sentiment_dict['pos'] > sentiment_dict['neg'] :
        return "Positive"

    elif sentiment_dict['neg'] > sentiment_dict['pos'] :
        return "Negative"

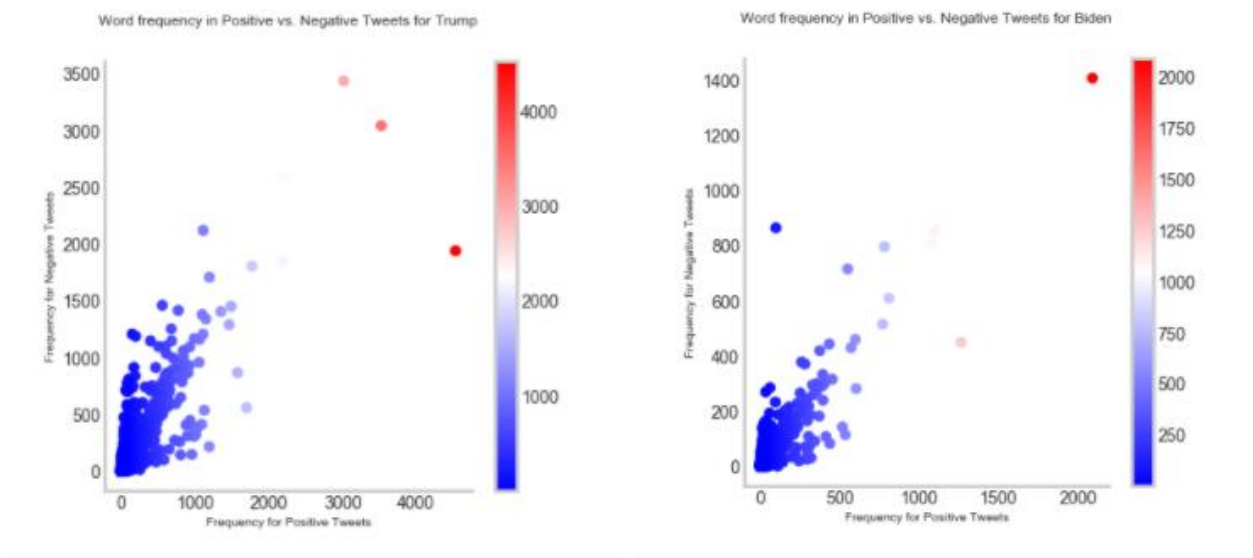
    else :
        return "Neutral"
```

We compared the average sentiment scores and looked at the top 10 words in each sentiment category.

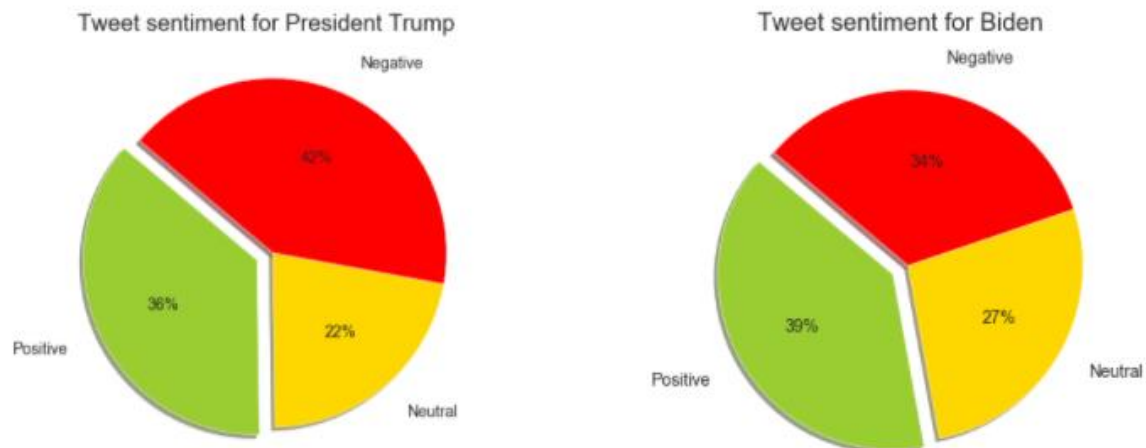


Twitter Sentiment Prediction Model Use Cases

Also, we looked positive and negative word frequency for each candidate.



As tweet volume for President Trump is much higher than former Vice President Joe Biden, we looked at % sentiment using pie chart.

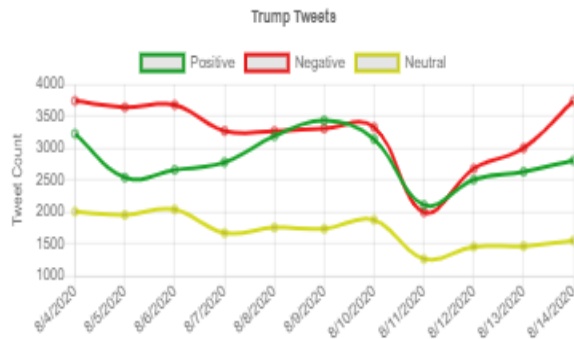


A huge part of Twitter conversation revolves around news and politics. That makes it an excellent place to measure public opinion, especially during election campaigns.

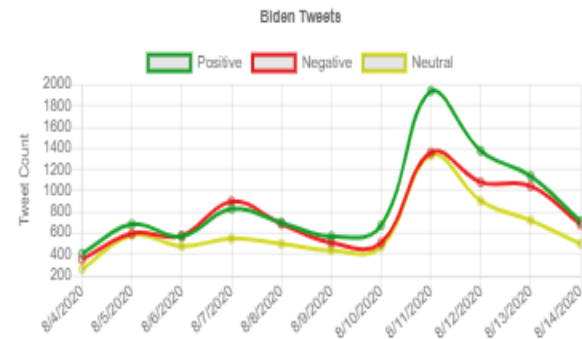
Twitter Sentiment Prediction Model Use Cases

First, we were able to count the number of positive and negative mentions for each candidate during a period of time. Following graph shows Trump and Biden's tweets based on sentiment. Joe Biden announced his VP choice on August 11 and one can see user sentiment to the news.

President Donald Trump

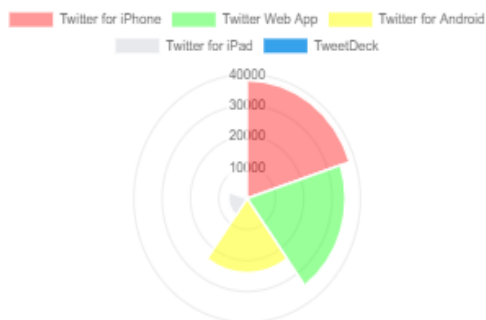


Former Vice President Joe Biden

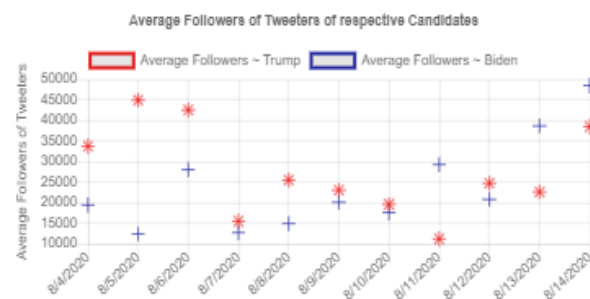


Interesting to understand the twitter user, followers and devices used to tweet the message.

OS/Devices of Tweet Sources



Tweet publisher's follower count



Machine Learning Model (AAHA Model)

Machine learning models can do pretty amazing things, even when it comes to something as subjective as the human language. With a machine learning, it's simple to get started with sentiment analysis of the Twitter data.

We compared Logistic Regression, Naive Bayes, and Support Vector Machines learning algorithms to build the review predictions. Final approach that does a combined "vote" of all three models.

Since we have limited data, we used cross-validation to split the data 10 ways and measure accuracy in an unbiased way.

```
# Calculate cross validation score
cat = ["Positive", "Negative", "Neutral"]
def calculate_cv(X, y):
    results = {
        'lr': [],
        'svm': [],
        'nb': [],
        'combined': []
    }
    lm = LogisticRegression()
    svm = LinearSVC()
    nb = MultinomialNB()
    vc = VotingClassifier([('lm', lm), ('svm', svm), ('nb', nb)])
    for c in cat:
        y_adj = np.array(y==c)
        results['lr'].append((cross_val_score(lm, X, y_adj, cv=10, scoring='accuracy').mean(), c))
        results['svm'].append((cross_val_score(svm, X, y_adj, cv=10, scoring='accuracy').mean(), c))
        results['nb'].append((cross_val_score(nb, X, y_adj, cv=10, scoring='accuracy').mean(), c))
        results['combined'].append((cross_val_score(vc, X, y_adj, cv=10, scoring='accuracy').mean(), c))
    return results

# Call function
cv_scores = calculate_cv(X_test, y_test)

# print model accuracy predictions

print("Model accuracy predictions\n")
for m,s in cv_scores.items():
    for ss in s:
        print("{M} model ({R} rating): {S:.1%}".format(M=m.upper(), R=ss[1], S=ss[0]))
    print()
```

After evaluating all the models, we selected **Logistic Regression** model. There are three models that are build for each category: Positive, Negative and Neutral.

Twitter Sentiment Prediction Model Use Cases

Training the model of choice:

- For training models, data was split 70%/30% where 30% will be used for prediction to gauge final accuracy of the model.
- For each review, the model that scores the highest will tell us which kind of review it likely is.

```
❏ # Split data
X_train, X_test, y_train, y_test = train_test_split(tfidf_d, df_pred['Sentiment'], test_size=0.3)
```

```
❏ # Build linear regression model and fit
def get_lr(x, y):
    models = []
    for c in cat:
        y_adj = np.array(y==c)
        lm = LogisticRegression()
        lm_f = lm.fit(x, y_adj)
        models.append(lm_f)
    return models

lr_m = get_lr(X_train, y_train)
```

Test output:

- Built a test function which allows to supply review to see how well the model will predict it's rating.
- For simplicity of testing, function allow only one review at a time.
- The program uses the stored TFIDF matrix to tokenize and transform our new review which is then fed to all three of logistic regression models.
- Each model has an independent assessment of how likely it is that our review is a positive hit.

```
❏ # Function to test sentiment
def test_sentiment(text,model):
    test_str = [text]
    test_new = tfidf_m.transform(test_str)

    print('Tweet text: "{R}"\n'.format(R=test_str[0]))
    print('Model Prediction')
    for m in range(0,3):
        print('Model ({M}): {P:.1%}'.format(M=cat[m], P=model[m].predict_proba(test_new)[0][1]))
```

Twitter Sentiment Prediction Model Use Cases

```
▶ # Good sentiment
test_sentiment('President Trump kept his word on trade policies. He is great for the businesses.',lr_m)

Tweet text: "President Trump kept his word on trade policies. He is great for the businesses."

Model Prediction
Model (Positive): 91.6%
Model (Negative): 2.3%
Model (Neutral): 10.1%
```

Save Model:

```
▶ import pickle
model_file_name = os.path.join("../Resources/model","final_model.pickle")
tfidf_model_file_name = os.path.join("../Resources/model","tfidf_model.pickle")

▶ # Save the model
pickle.dump(lr_m, open(model_file_name, 'wb'))

▶ # Save tfd
pickle.dump(tfidf_m, open(tfidf_model_file_name, "wb"), protocol=0)
```

Loading the model:

```
# Loading the model file
loaded_model = pickle.load(open(model_file_name, 'rb'))
tfidf_model = pickle.load(open(tfidf_model_file_name, "rb"))

# Transform input value
pred_data = tfidf_model.transform([text])

# Predict sentiment - 3 models for 3 sentiments
positive = (float("{:.1f}".format(loaded_model[0].predict_proba(pred_data)[0][1])) * 100)
negative = (float("{:.1f}".format(loaded_model[1].predict_proba(pred_data)[0][1])) * 100)
neutral = (float("{:.1f}".format(loaded_model[2].predict_proba(pred_data)[0][1])) * 100)
```


```
if positive > 50:
    sentText = "Positive"
elif negative > 50:
    sentText = "Negative"
else:
    sentText = "Neutral"

# Build a dictionary to return values
prediction = {"Positive": positive, "Negative": negative, "Neutral": neutral}
```

Twitter Sentiment Prediction Model Use Cases


Model Testing:

We build user interface screen to test AAHA Twitter Sentiment model that predicts user entered tweet sentiment. Results are compared with Google NLP model, Vader Model and TextBlob model.

**Twitter Sentiment Analysis**

Home Visualization Prediction Documentation References Contact

ML Model - Predict Tweet Sentiment



Sentiment Text: People feel President Trump should have shut the country in the early stage. He killed too many people because his COVID19 policies

Confidence :	80.0%	AAHAA Model Sentiment:	Negative
Score :	-80.0%	Google NPL Sentiment:	Negative
Magnitude :	1.6%		
Positive :	0.0%	VADER Model Sentiment:	Negative
Netural :	82.4%		
Negative :	17.6%		
Compound :	-67.05%		
Polarity :	13.33%	TextBlob Model Sentiment:	Positive
Subjectivity :	26.67%		

Conclusion

While our data set was smaller, we were surprised at just how much insight could be drawn from this data. There are clear, explainable differences between the sentiments for two political leaders.

If anything, this shows that more and more, Twitter is reflective of a politician's *true* beliefs, and sometimes is more revealing than any other source.

AAHAA Model gave fairly same results across industry sentiment analysis API's.

Twitter Sentiment Prediction Model Use Cases

References

References

[scikit-learn for NLP](#)
[Working With Text Data — scikit-learn 0.23.2 documentation](#)

[VADER Sentiment Analysis](#)
[Using VADER to handle sentiment analysis with social media text](#)
VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool.

[TextBlob Sentiment Analysis](#)
[TextBlob Sentiment: Calculating Polarity and Subjectivity](#)
Simple, Pythonic, text processing--Sentiment analysis, part-of-speech tagging, noun phrase extraction, translation, and more

[AFINN Sentiment Analysis](#)
[AFINN sentiment analysis in Python: Wordlist-based approach for sentiment analysis](#)
[AFINN sentiment analysis in Python](#)

[Google NLP Sentiment Analysis](#)
Google NLP API has several methods for performing analysis and annotation on text. Sentiment analysis inspects the given text and identifies the prevailing emotional opinion within the text, especially to determine a writer's attitude as positive, negative, or neutral. Sentiment analysis is performed through the [analyze sentiment method](#).
[AFINN sentiment analysis in Python](#)

[NLTK Corpus](#)
The [nltk.corpus](#) package defines a collection of corpus reader classes to access the contents of a diverse set of corpora
[How to Clean Text for Machine Learning with Python](#)

[Research Papers](#)
[A Review towards the Sentiment Analysis Techniques for the Analysis of Twitter Data](#)
[Sentiment Analysis for Social Media 2020](#)

Data Sources

[Twitter Developer Labs/API Reference](#)

Credits

- [Medium.com](#) articles
- [Towardsdatascience.com](#) articles
- Dimitrios Effrosynidis (Text Cleaning Techniques)
- Twitter Developer Page
- Paul Arias (RU Bootcamp Instructor)

Team Members

- Ajay Patil
- Akshita Parasrampur
- Hasti Patel
- Archana Ashish
- Aslam Momin

Declaration

The authors declare and solemnly affirm that this research has neither been funded by any political or religious groups nor are the authors in any way affiliated to any institutions with direct or indirect access to groups with biased interests. This research work has been carried out in the interests of technology and academics.