Sentiment Analysis on Twitter

Andre Silva

November 18, 2020

1. Introduction

1.1 Background

Focused on dynamic comment replies and interactions, Twitter has become one of the most popular platforms where people can express feelings and thoughts on several different topics. With more than 340 million monthly active users, Twitter timelines are filled with tweets about politics, sports, games, celebrity gossip and general news, providing tons of valuable feedback and reviews, which is highly useful to companies and governments.

1.2 Business Problem

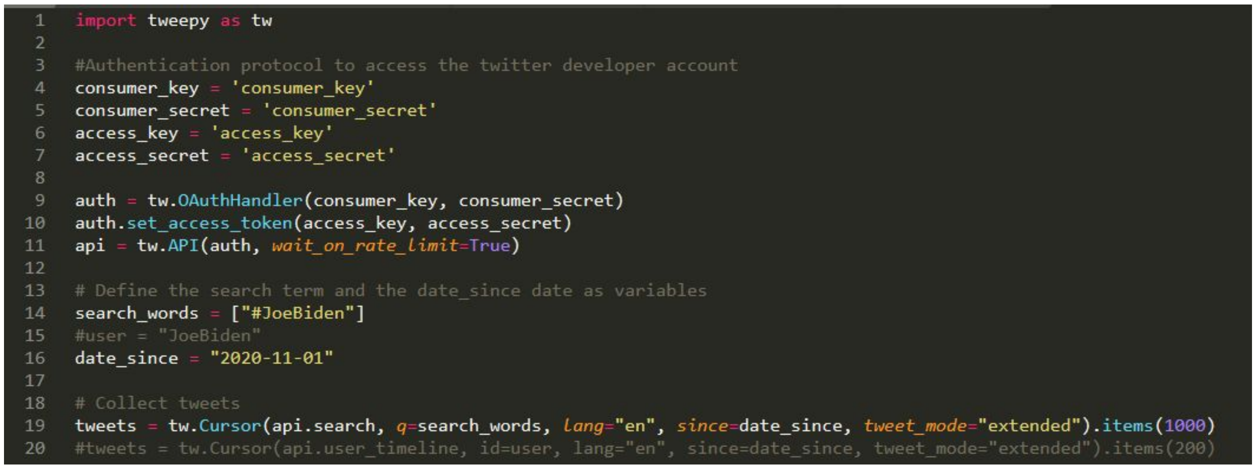
Public opinion in large scale is vital to several different scenarios within our modern society. In order to ensure the best release of a new product, to measure the impact of a new government policy, to understand the public opinion toward an specific topic or to undermine preference of different profiles of clients, all require a solid system to measure what the general public thinks about the subject. However, gathering all this data can be challenging since it is hard to measure the impact of something on a large sample of people before releasing the actual product or news. Fortunately, there are platforms around the internet where a large number of users provide free and abundant thoughts all day long, the larger one of these platforms being Twitter.

This article aims to provide a helpful methodology to collect large numbers of tweets from specific users of hashtags and then, using a decision tree machine learning algorithm, classify them in positive, negative or neutral, providing an overall panorama of the sentiment users have. In this case, as an example, we are going to analyse the sentiments of Twitter users regarding the US election of 2020, by collecting tweets from #USElections2020, #JoeBiden and #DonaldTrump.

2. Data

2.1 Data collection

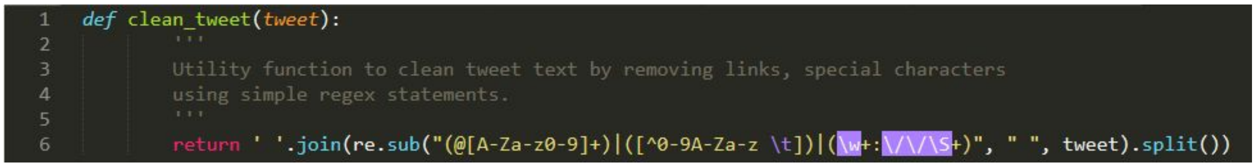
For the data collection I used the Twitter API for developers which allows tweets scraping sorted by hashtag, username or date. For the results on this paper I collected a thousand tweets from each tag (#USElections2020, #JoeBiden and #DonaldTrump).



```python
import tweepy as tw

#Authentication protocol to access the twitter developer account
consumer_key = 'consumer_key'
consumer_secret = 'consumer_secret'
access_key = 'access_key'
access_secret = 'access_secret'

auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tw.API(auth, wait_on_rate_limit=True)

# Define the search term and the date_since date as variables
search_words = ["#JoeBiden"]
#user = "JoeBiden"
date_since = "2020-11-01"

# Collect tweets
tweets = tw.Cursor(api.search, q=search_words, lang="en", since=date_since, tweet_mode="extended").items(1000)
#tweets = tw.Cursor(api.user_timeline, id=user, lang="en", since=date_since, tweet_mode="extended").items(200)
```

2.2 Data cleaning

After collecting the tweets I used simple regex syntax in python to remove links, special characters and overall data that's not useful to undermine the sentiment.



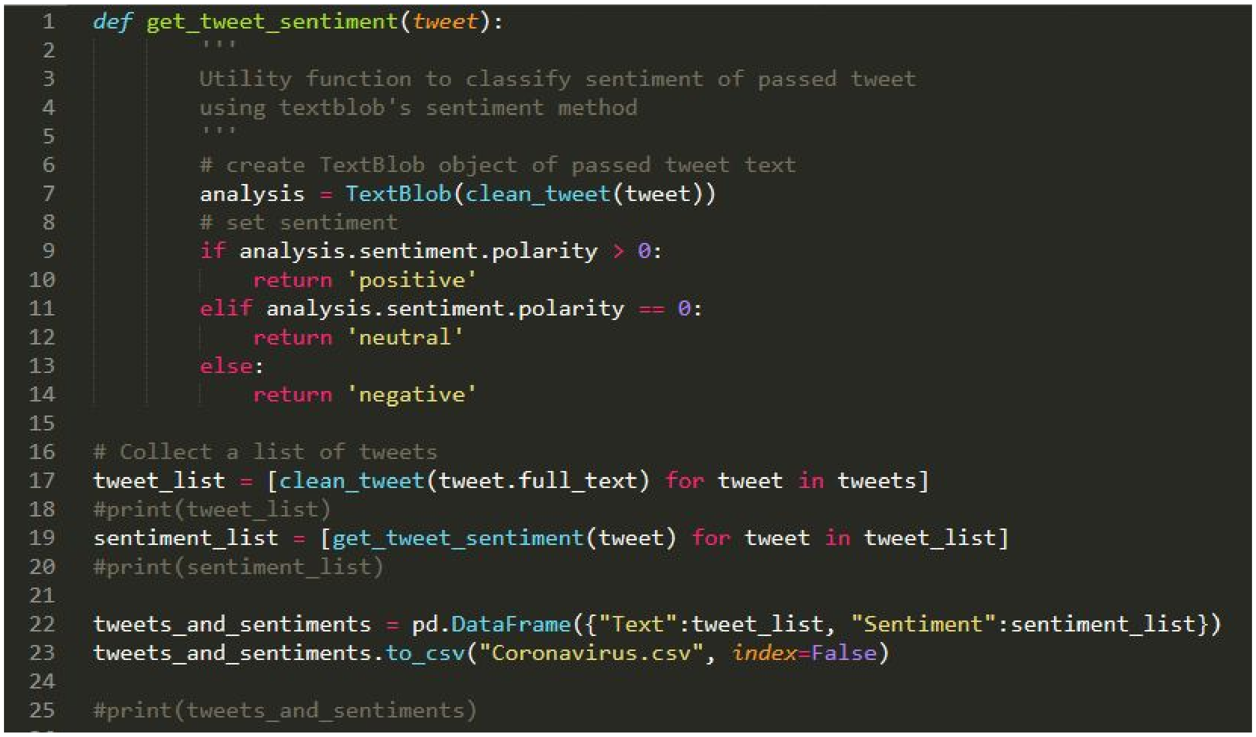
```python
def clean_tweet(tweet):
        '''
        Utility function to clean tweet text by removing links, special characters
        using simple regex statements.
        '''
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())
```

3. Methodology

3.1 Getting the Sentiment

To get the sentiment of each tweet I used a library to process textual data called TextBlob, which has useful classifications and features for most words in the english dictionary. For every word in the tweet, a polarity value of negative or positive sentiment gets assigned to that word, then, based on the polarity of all the words in the sentence, a general polarity gets assigned to the entire sentence. It is important to notice that some words are assigned as neutral because they don't convey any specific sentiment, and also some words might carry either negative or positive sentiments based on context. That's why the polarity value sometimes might not be strong enough to either side, so instead of being negative or positive, the sentence gets classified as neutral.

That doesn't mean however that neutral sentences convery no value for the analysis of the topic or profile on Twitter. If a general topic is being treated as neutral by the majority of users, this might indicate that people don't have a strong opinion on the subject based on current information, or that the public is just indifferent to the topic, which might be a useful insight depending on the business problem. Also, neutral tweets might be spam tweets promoting a profile or product, using the tags to get more views, which contributes nothing to the business problem, so it's useful to create a third category for classifying neutral sentiments.

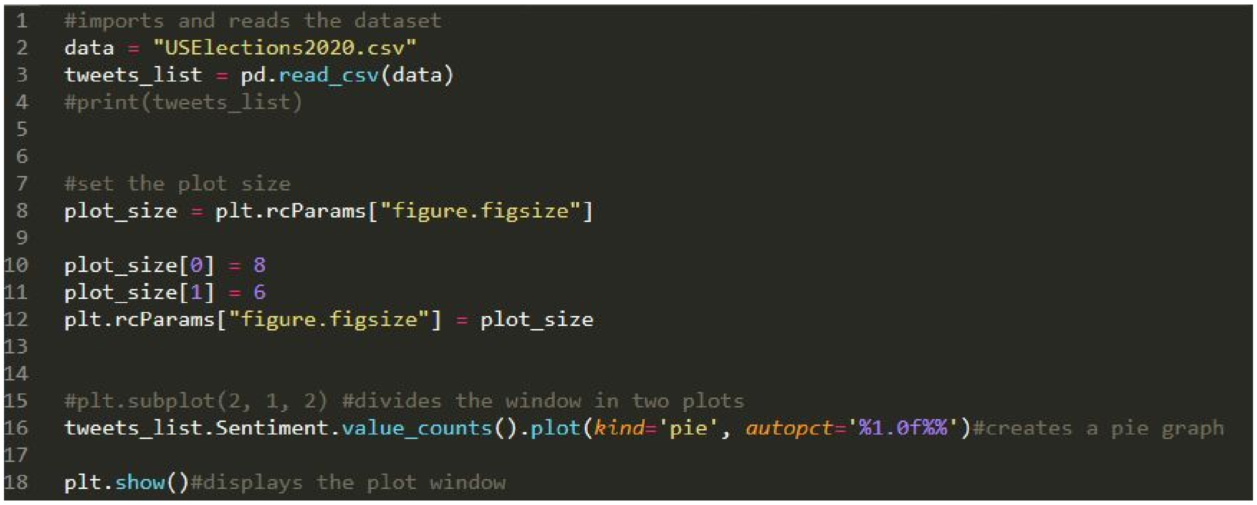


```python
def get_tweet_sentiment(tweet):
        '''
        Utility function to classify sentiment of passed tweet
        using textblob's sentiment method
        '''
        # create TextBlob object of passed tweet text
        analysis = TextBlob(clean_tweet(tweet))
        # set sentiment
        if analysis.sentiment.polarity > 0:
            return 'positive'
        elif analysis.sentiment.polarity == 0:
            return 'neutral'
        else:
            return 'negative'

# Collect a list of tweets
tweet_list = [clean_tweet(tweet.full_text) for tweet in tweets]
#print(tweet_list)
sentiment_list = [get_tweet_sentiment(tweet) for tweet in tweet_list]
#print(sentiment_list)

tweets_and_sentiments = pd.DataFrame({"Text":tweet_list, "Sentiment":sentiment_list})
tweets_and_sentiments.to_csv("Coronavirus.csv", index=False)

#print(tweets_and_sentiments)
```

3.2 Visualization and measuring results

Be mindful that the results give us insight on the sentiments based on the tweet alone, disregarding context such as others tweets or things that happened in the real world outside of the internet. Tweets like news headlines from informative profiles usually have a very neutral language, being interpreted as positive or negative depending on who reads it.

For displaying the results I plot two graphs using standard data visualizations features of the MatplotLib library, a very common python library for data science projects.

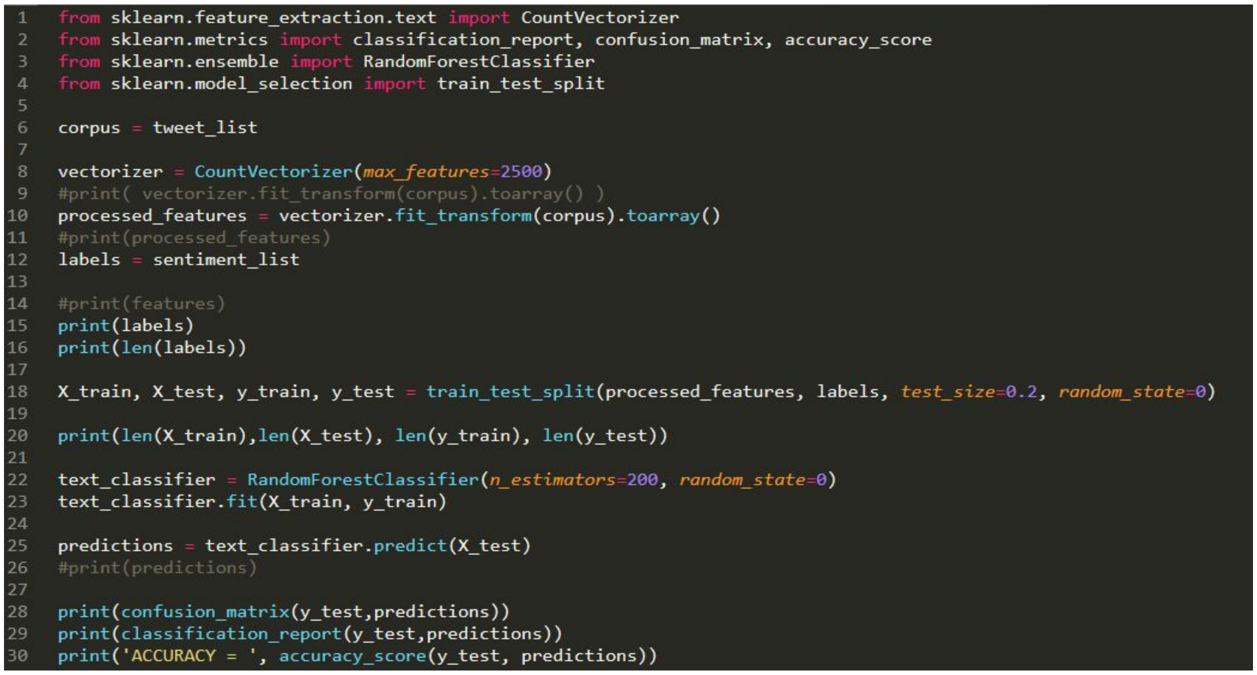


```python
#imports and reads the dataset
data = "USElections2020.csv"
tweets_list = pd.read_csv(data)
#print(tweets_list)


#set the plot size
plot_size = plt.rcParams["figure.figsize"]

plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size


#plt.subplot(2, 1, 2) #divides the window in two plots
tweets_list.Sentiment.value_counts().plot(kind='pie', autopct='%1.0f%%')#creates a pie graph

plt.show()#displays the plot window
```

3.3 Predicting tweet sentiment

Once the initial collection is over, we now have a cleaned and labeled data frame with three thousand tweets to future machine learning analysis. The possibilities for future models and insights with this data are endless, so I am going to introduce and focus on one exemple only.

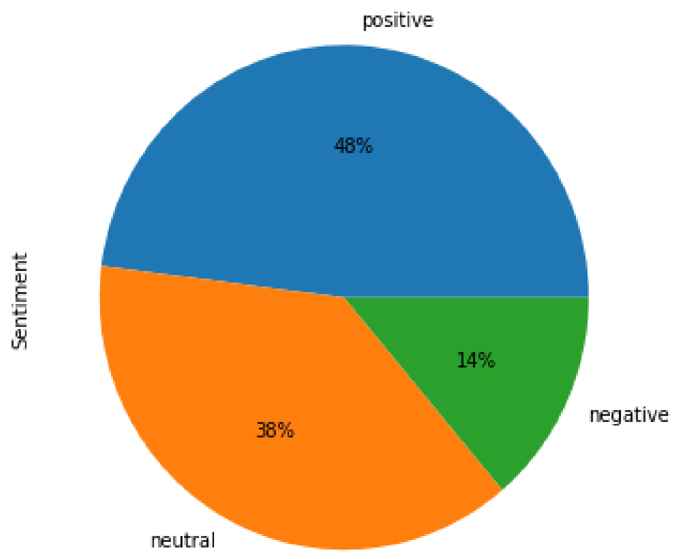
I applied a decision tree algorithm to predict the sentiment of a tweet. In natural language processing, having your data more focused on the problem you are trying to solve is highly beneficial. Depending on the type of text you're trying to analyse, some words or sentences might appear in a certain way more often. That's why we are collecting and training a natural language predictor to further increase the accuracy of the project as a whole, including the first classification made by TextBlob.

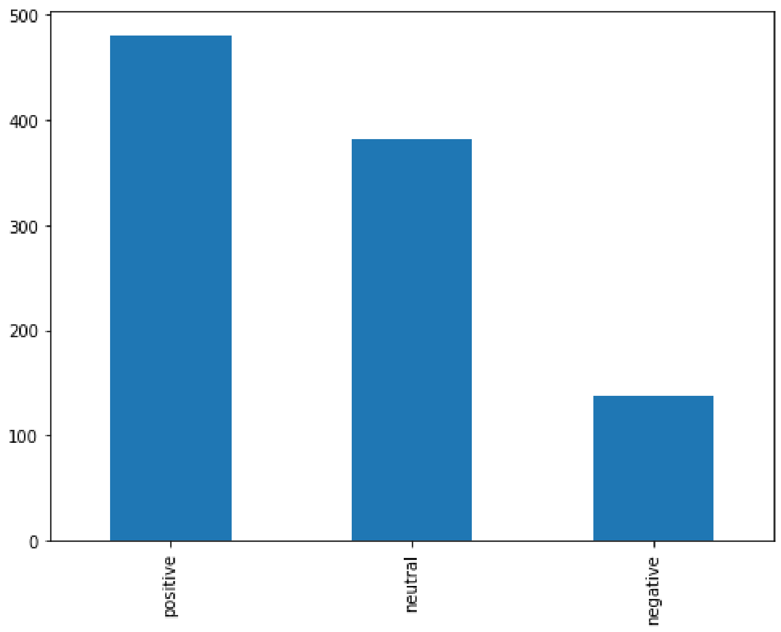


```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

corpus = tweet_list

vectorizer = CountVectorizer(max_features=2500)
#print( vectorizer.fit_transform(corpus).toarray() )
processed_features = vectorizer.fit_transform(corpus).toarray()
#print(processed_features)
labels = sentiment_list

#print(features)
print(labels)
print(len(labels))

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state=0)

print(len(X_train),len(X_test), len(y_train), len(y_test))

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)

predictions = text_classifier.predict(X_test)
#print(predictions)

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print('ACCURACY = ', accuracy_score(y_test, predictions))
```

4. Results

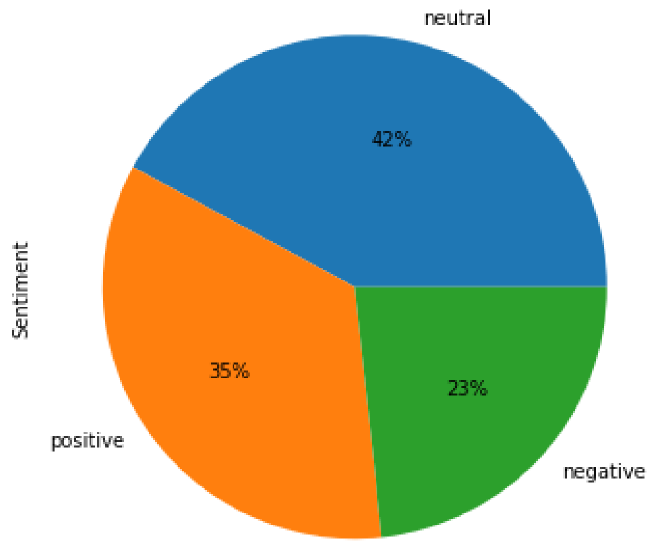
4.1 #USElections20202 Results

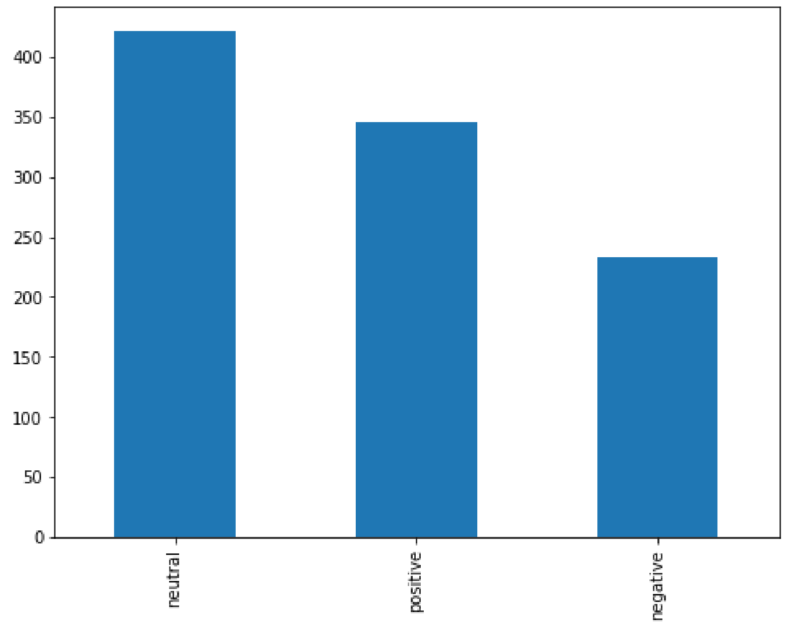




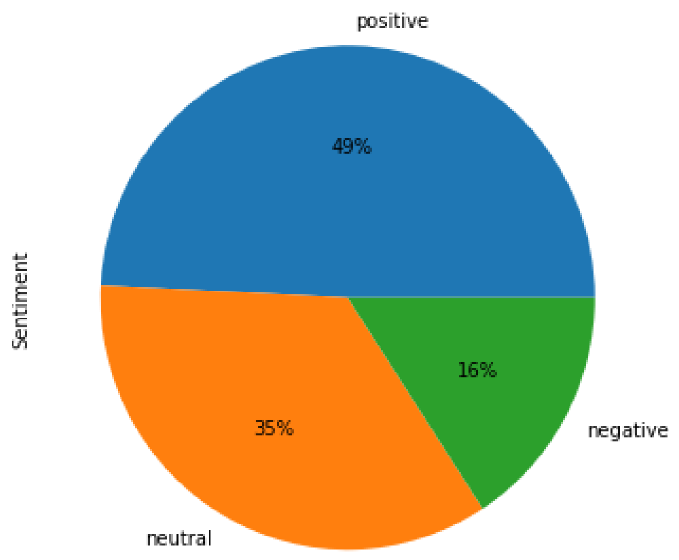
These tweets were collected one week after Joe Biden’s win in the US elections. Here we see that most of the tweets on this topic have been positive, while neutral tweets (mostly news reports) also take a large percentage. Now let’s look at the results for each candidate tag separately.

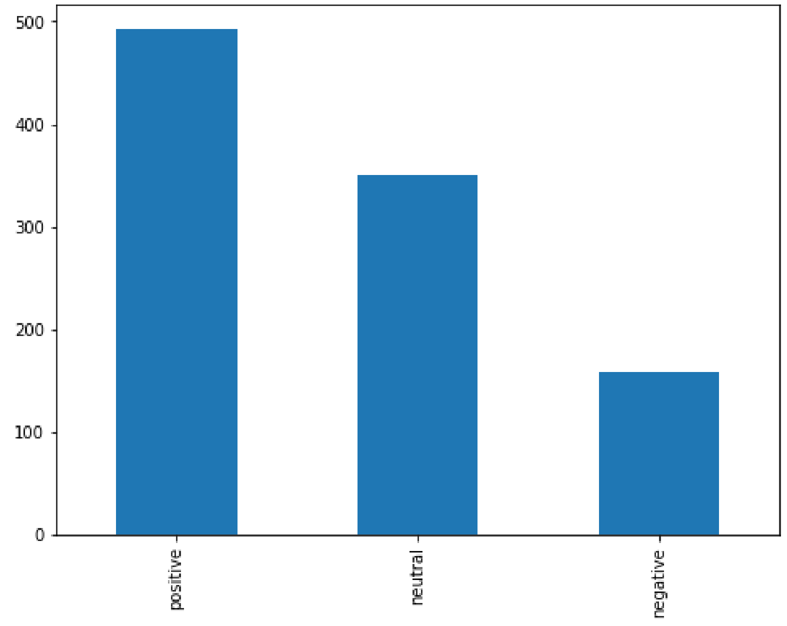
4.2 #DonaldTrump Results





4.3 #JoeBiden Results





4.4 Predicting Sentiments Results

	precision	recall	f1-score	support
negative	0.91	0.36	0.51	28
neutral	0.80	0.93	0.86	80
positive	0.83	0.87	0.85	92
accuracy			0.82	200
macro avg	0.85	0.72	0.74	200
weighted avg	0.83	0.82	0.81	200

ACCURACY = 0.82

Applying the decision tree algorithm with sklearn metrics library I was able to achieve a 82% accuracy in predicting the sentiment of new tweets. As the table shows through the f1-score measurement, negative tweets are harder to classify, while positive and neutral tweets are predicted with higher accuracy.

The above table is generated by the sklearn classification report, which displays a few useful metrics from the prediction model.

5. Discussion

As stated before, natural language processing involves many inaccuracies regarding the context of language and vocabulary. Future development in this project might bring more insights on the relationship between certain words in sentences that undermines a sentiment.

Furthermore, on the next step of this project more sentiments might be added besides only positive and negative. Sentiments like happiness, anger, uncertainty, excitement or fear might provide even more valuable insights on users sentiments on social media.

A popular method to increase the accuracy of machine learning models is having more data to process. I ran this project on a free license for a jupyter notebook in a virtual machine, which limits much of the data processing available. I only analysed a thousand tweets per tag, which consists only of tweets made in a interval of a couple hours. By collecting a larger data sample it is possible to get insights from users over much larger intervals of time, resulting in more valuable and accurate conclusions. More data also contributes for a more polished machine learning model for predicting the tweets sentiment, increasing the effectiveness of the project as a whole.

6. Conclusion

In this project I used the Twitter Developer API, Natural Language Processing and Machine Learning Decision Tree Classification algorithm to analyse Twitter users sentiments on the topic of the US election of 2020, as well as the two candidates. Getting feedback and sentiment analysis from users on social media provides valuable information for decision making in various fields, therefore, by massively implementing such algorithms in different social medias, these platforms might become the main source for collecting such valuable information.

Applying basic machine learning with limited resources it is possible to extract very useful insights from the data. With further development and research, this framework might bear even more useful and accurate insights in basically any scenario where fast, cheap and abundant data from popular opinion is required.