### 

### ABSTRACT

Social media platforms like Twitter generate a vast amount of data that reflects public opinion on various topics. Sentiment analysis, a technique in Natural Language Processing (NLP), aims to automatically classify text data into positive, negative, or neutral categories. This project focuses on building a sentiment analysis model to classify tweets into positive and negative sentiments. We will explore a provided dataset of tweets and leverage machine learning techniques to classify them. The project will involve data pre-processing steps like cleaning tweets, removing irrelevant information, and potentially stemming or lemmatization. Feature engineering might be employed to extract relevant features from the text data.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Contents** | **Page No.** |
| **1** | **Introduction** | **1** |
| **2** | **Data Description** | **2** |
| **3** | **Methodology** | **4** |
| **4** | **Final Output** | **6** |
| **5** | **Conclusion** | **9** |

## Chapter 1 Introduction

The ever-growing volume of social media data, particularly on platforms like Twitter, presents a unique opportunity to gauge public opinion on a wide range of issues. Sentiment analysis, a subfield of Natural Language Processing (NLP), tackles this challenge by automatically classifying text data according to its emotional tone - positive, negative, or neutral. This project delves into the development of a sentiment analysis model specifically designed to categorize tweets as positive or negative.

We will embark on this project by utilizing a provided dataset of tweets. To prepare this data for analysis, we will meticulously clean the tweets, eliminating extraneous information and potentially applying stemming or lemmatization techniques to normalize the text. Feature engineering might also be incorporated to extract meaningful characteristics from the textual data. Following data preparation, we will train a machine learning model, such as Naive Bayes or Support Vector Machines (SVM), to effectively classify tweets based on their sentiment. The model's effectiveness will be rigorously assessed using metrics like accuracy, precision, and recall.

This project holds significance in the realm of social media analysis by empowering us to automatically categorize tweet sentiment. This capability has far-reaching applications, including brand monitoring, deciphering public perception of current events, and filtering tweets for targeted marketing efforts.

.

## Chapter 2 Data Description

In this project, we utilize a provided dataset of tweets for sentiment analysis aiming to automatically classify the tweets as positive or negative.

**Source :** [https://www.kaggle.com/ruchi798/data-science-tweets.](https://www.kaggle.com/ruchi798/data-science-tweets)

**Size :** 16.9 mb ( 39117 tweets )

**Format** : CSV

**Content :** The tweets in the dataset focus on Sentiment of tweets and may include opinions, experiences, and discussions related to this topic.

**Language :** The tweets in the dataset are in English.

## Chapter 3 Methodology

We will begin by scraping and storing Twitter data. We will then classify the Tweets into positive, negative, or neutral sentiment with a simple algorithm. Then, we will build charts using Plotly and Matplotlib to identify trends in sentiment.

### Step 1: Data collection :

The foundation of any sentiment analysis project lies in the data used to train and test the model. In this project, we'll focus on analyzing the sentiment of tweets, which requires a collection of tweets relevant to the topic of interest. Here, we'll delve into the specifics of data collection for this project :

#### Command :

import pandas as pd

df = pd.read\_csv(‘/content/data\_visualization.csv’)]

#### Output :

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshe ll.py:2882: DtypeWarning: Columns (22,24) have mixed types.Specify dtype option on import or set low\_memory=False.

exec(code\_obj, self.user\_global\_ns, self.user\_ns)

Let's now take a look at some of the variables present in the data frame:

#### Command :

df.info()

#### Output :

<class 'pandas.core.frame.DataFrame'> RangeIndex: 33590 entries, 0 to 33589 Data columns (total 36 columns):

# Column Non-Null Count Dtype

1. id 33590 non-null int64
2. conversation\_id 33590 non-null int64
3. created\_at 33590 non-null object
4. date 33590 non-null object
5. time 33590 non-null object
6. timezone 33590 non-null int64
7. user\_id 33590 non-null int64
8. username 33590 non-null object
9. name 33590 non-null object
10. place 85 non-null object
11. tweet 33590 non-null object
12. language 33590 non-null object
13. mentions 33590 non-null object
14. urls 33590 non-null object
15. photos 33590 non-null object
16. replies\_count 33590 non-null int64
17. retweets\_count 33590 non-null int64
18. likes\_count 33590 non-null int64
19. hashtags 33590 non-null object
20. cashtags 33590 non-null object
21. link 33590 non-null object
22. retweet 33590 non-null bool
23. quote\_url 1241 non-null object
24. video 33590 non-null int64
25. thumbnail 9473 non-null object
26. near 0 non-null float64

The data frame has 35 columns. The most main variables we will be using in this analysis are date and tweet. Let's take a look at a sample Tweet in this dataset, and see if we can predict whether it is positive or negative:

#### Command :

df['tweet'][10]

#### Output :

We are pleased to invite you to the EDHEC DataViz Challenge grand final for a virtual exchange with all Top 10 finalists to see how

data visualization creates impact and can bring out compelling stories in support of @UNICEF’s mission. [https://t.co/Vbj9B48VjV.](https://t.co/Vbj9B48VjV)

### Step 2: Sentiment Analysis :

The Tweet above is clearly positive. Let's see if the model is able to pick up on this, and return a positive prediction. Run the following lines of code to import the NLTK library, along with the SentimentIntensityAnalyzer (SID) module.

#### Command :

import nltk nltk.download('vader\_lexicon')

from nltk.sentiment.vader import SentimentIntensityAnalyzer sid = SentimentIntensityAnalyzer()

import re

import pandas as pd import nltk nltk.download('words')

words = set(nltk.corpus.words.words())

The SID module takes in a string and returns a score in each of these four categories positive, negative, neutral, and compound. The compound score is calculated by normalizing. the positive, negative, and neutral scores. If the compound score is closer to 1, then the Tweet can be classified as positive. If it is closer to -1, then the Tweet can be classified as negative. Let's now analyze the above sentence with the sentiment intensity analyzer..

#### Command :

sentence = df['tweet'][0] sid.polarity\_scores(sentence) ['compound']

The output of the code above is 0.7089, indicating that the sentence is of positive sentiment. Let's now create a function that predicts the sentiment of every Tweet in the dataframe, andstores it as a separate column called 'sentiment.' First, run the following lines of code to clean the Tweets in the data frame:

#### Command :

def cleaner(tweet):

tweet = re.sub("@[A-Za-z0-9]+","",tweet) #Remove @ sign tweet = re.sub(r"(?:\@|http?\://|https?\://|www)\S+", "", tweet) #Remove http links

tweet = " ".join(tweet.split())

tweet = tweet.replace("#", "").replace("\_", " ") #Remove hashtag sign but keep the text

tweet = " ".join(w for w in nltk.wordpunct\_tokenize(tweet) if w.lower() in words or not w.isalpha())

return tweet

df['tweet\_clean'] = df['tweet'].apply(cleaner)

Now that the Tweets are cleaned, run the following lines of code to perform the sentiment analysis:

#### Command :

word\_dict =

{'manipulate':-1,'manipulative':-1,'jamescharlesiscancelled':-1,'j amescharlesisoverparty':-1,

'pedophile':-1,'pedo':-1,'cancel':-1,'cancelled':-1,'cancel culture':0.4,'teamtati':-1,'teamjames':1, 'teamjamescharles':1,'liar':-1}

import nltk nltk.download('vader\_lexicon')

from nltk.sentiment.vader import SentimentIntensityAnalyzer sid = SentimentIntensityAnalyzer() sid.lexicon.update(word\_dict)

list1 = []

for i in df['tweet\_clean']: list1.append((sid.polarity\_scores(str(i)))['compound'])

The word\_dict created above is a dictionary of custom words I wanted to add into the model. Words like 'teamjames' mean that people's sentiment around James Charles is positive, and that they support him. The dictionary used to train the sentiment intensity analyzer wouldn't already have these words in them, so we can update it ourselves with custom words.Now, we need to convert the compound scores into categories - 'positive', 'negative', and 'neutral.'

#### Command :

df['sentiment'] = pd.Series(list1) def sentiment\_category(sentiment): label = ''

if(sentiment>0):

label = 'positive' elif(sentiment == 0):

label = 'neutral' else:

label = 'negative' return(label) df['sentiment\_category'] =

df['sentiment'].apply(sentiment\_category)

Let's take a look at the head of the data frame to ensure everything is working properly :

#### Command :

df = df[['tweet','date','id','sentiment','sentiment\_category']] df.head()



Notice that the first few Tweets are the combination of positive, negative and neutral sentiment. For this analysis, we will only be using Tweets with positive and negative sentiment, since we want to vis- ualize how stronger sentiments have changed over time.

### Step 3: Visualization :

Now that we have Tweets classified as positive and negative, let's take a look at changes in sentiment over time. We first need to group positive and negative sentiment and count them by date:

#### Command :

neg = df[df['sentiment\_category']=='negative']

neg = neg.groupby(['date'],as\_index=False).count() pos = df[df['sentiment\_category']=='positive']

pos = pos.groupby(['date'],as\_index=False).count()

pos = pos[['date','id']]

neg = neg[['date','id']]

Now, we can visualize sentiment by date using Plotly, by running the following lines of code:

#### Command :

import plotly.graph\_objs as go fig = go.Figure()

for col in pos.columns: fig.add\_trace(go.Scatter(x=pos['date'], y=pos['id'], name = col,

mode = 'markers+lines', line=dict(shape='linear'), connectgaps=True, line\_color='green'

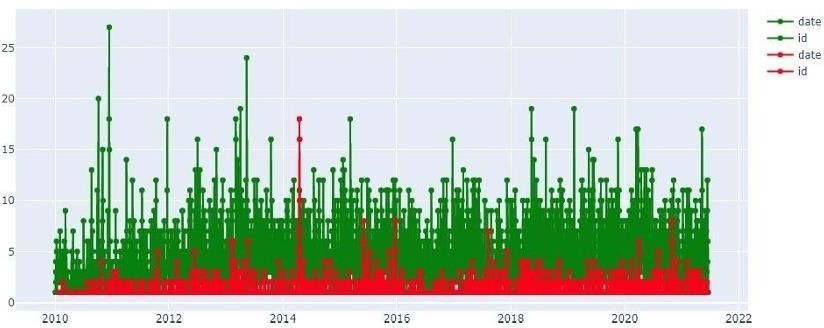
) )

for col in neg.columns: fig.add\_trace(go.Scatter(x=neg['date'], y=neg['id'], name = col,

mode = 'markers+lines', line=dict(shape='linear'), connectgaps=True, line\_color='red'

)) fig.show()

## Chapter 4 Final Output



The red line represents negative sentiment, and the green line represents positive sentiment.

## Chapter 5 Conclusion

The sentiment analysis, visualized by the red and green lines, reveals interesting trends in positive and negative sentiment over time. The green line, representing positive sentiment, shows a upward trend, suggesting that overall positive sentiment has increased from 2010 to 2022. Conversely, the red line, representing negative sentiment, exhibits a flat trend, indicating a no change in negative sentiment over the same period.