sentimentanalysis

May 7, 2023

Name: Manasa Kinnera

email: mxk51980@ucmo.edu

[ ]:
```
# Install Libraries
!pip install textblob
!pip install tweepy
!pip install pycountry
!pip install wordcloud
!pip install langdetect
!pip install pycountry
```

[22]:
```
from textblob import TextBlob
import sys
import tweepy
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os
import nltk
import pycountry
import re
import string
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from langdetect import detect
from nltk.stem import SnowballStemmer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import CountVectorizer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/prasku/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

[22]:
```
True
```

What is sentiment analysis

Sentiment analysis is a computational method that allows businesses to automatically extract and quantify subjective information from text data, including social media posts, customer reviews, and feedback forms. By analyzing the emotional tone of the text, sentiment analysis can determine whether the sentiment is positive, negative, or neutral, and evaluate its impact on the overall message's meaning.

Libraries used in Sentiment Analysis

There are several libraries and tools available for performing sentiment analysis. Tweepy is a Python library that enables access to the Twitter API, allowing businesses to retrieve tweets based on a specific query. While Tweepy does not have built-in sentiment analysis capabilities, it can be used in conjunction with other sentiment analysis libraries.

Word cloud is another tool used in sentiment analysis to visually represent the most frequently used words in a given set of text data. By analyzing the frequency and importance of words, businesses can gain insights into the most commonly used words and phrases associated with a particular sentiment.

TextBlob is a popular Python library for performing sentiment analysis. It uses natural language processing techniques to analyze the polarity of a piece of text, returning a sentiment score ranging from -1 (negative sentiment) to 1 (positive sentiment). TextBlob also allows for the identification of subjectivity and objectivity in text, which can help businesses understand how opinions and emotions are expressed.

Businesses can use sentiment analysis to gain insights into customer sentiment and improve their marketing, customer service, and product development strategies. For example, sentiment analysis can help businesses identify common pain points and complaints from customers, as well as monitor the sentiment surrounding a new product or service launch. By analyzing customer sentiment in real-time, businesses can respond quickly and effectively to customer feedback, ultimately improving customer satisfaction and loyalty.

Introduction

In this code, we are authenticating with Twitter's API using Tweepy library. We will use this authentication to access Twitter data, in our case tweets containing "WALMART".

Analysis

To access Twitter data, we need to authenticate with Twitter's API using keys and access tokens. Tweepy library makes it easier to work with Twitter's API in Python. With this authentication, we can search for tweets containing certain keywords and perform sentiment analysis on the text.

Sample Data Explanation

In this code, we are using authentication keys and access tokens to access Twitter's API using Tweepy. We then use this authentication to generate comments on each line of the code.

consumerKey and consumerSecret are API keys provided by Twitter for authentication.

accessToken and accessTokenSecret are access tokens provided by Twitter for authentication.

auth = tweepy.OAuthHandler(consumerKey, consumerSecret) creates an OAuthHandler instance with the consumer key and consumer secret.

auth.set_access_token(accessToken, accessTokenSecret) sets the access token and access token secret on the OAuthHandler instance.

api = tweepy.API(auth) creates an API instance with the authenticated OAuthHandler instance. This instance can be used to interact with Twitter's API, including searching for tweets.

Overall, this code is setting up the authentication required to access Twitter's API using Tweepy, which we will use to search for tweets containing "WALMART" and perform sentiment analysis.

[6]:
```
# Authentication
consumerKey = "2CJbp8m7Rg26LtZEKAeVXWceG"
consumerSecret = "IbULWKlSOuvhLnMTcFkzvMjDfZUcBM6eVbdLYMUbNafhieX8Y6"
accessToken = "1119095264961126402-8QDKOFvTjGmV7w1hUtGsTyCyH77bNE"
accessTokenSecret = "unheMaBnZsHrcBt3mK1bE8Ib4wtE3tPtAwSA65YOzWIKO"

auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth)
```

[7]:
```
#Testing the credentials
try:
    api.verify_credentials()
    print("Authentication OK")
except Exception as e:
    print("Error during authentication")
    print(str(e))
```

```
Authentication OK
```

[8]:
```
# Import necessary libraries
from textblob import TextBlob
from nltk.sentiment import SentimentIntensityAnalyzer
import tweepy

# Define a function to calculate percentage
def percentage(part,whole):
    return 100 * float(part)/float(whole)

# Ask user for input keyword and number of tweets to analyze
keyword = input("Please enter keyword or hashtag to search: ")
noOfTweet = int(input ("Please enter how many tweets to analyze: "))

# Use Twitter API to fetch tweets containing the input keyword
tweets = tweepy.Cursor(api.search_tweets, q=keyword).items(noOfTweet)

# Initialize counters and lists
positive  = 0
negative = 0
neutral = 0
```

```
polarity = 0
tweet_list = []
neutral_list = []
negative_list = []
positive_list = []

# Loop through each tweet fetched from API
for tweet in tweets:
    # Append the tweet text to a list
    tweet_list.append(tweet.text)

    # Analyze the tweet sentiment using TextBlob
    analysis = TextBlob(tweet.text)

    # Analyze the tweet sentiment using SentimentIntensityAnalyzer
    score = SentimentIntensityAnalyzer().polarity_scores(tweet.text)

    # Extract the polarity scores
    neg = score['neg']
    neu = score['neu']
    pos = score['pos']
    comp = score['compound']

    # Calculate the overall polarity score
    polarity += analysis.sentiment.polarity

    # Classify the tweet as positive, negative or neutral based on the␣
 ↪sentiment scores
    if neg > pos:
        negative_list.append(tweet.text)
        negative += 1

    elif pos > neg:
        positive_list.append(tweet.text)
        positive += 1

    elif pos == neg:
        neutral_list.append(tweet.text)
        neutral += 1

# Calculate percentage of each sentiment category
positive = percentage(positive, noOfTweet)
negative = percentage(negative, noOfTweet)
neutral = percentage(neutral, noOfTweet)
polarity = percentage(polarity, noOfTweet)

# Format percentage values to one decimal place
```

```
positive = format(positive, '.1f')
negative = format(negative, '.1f')
neutral = format(neutral, '.1f')
```

The code above performs sentiment analysis on tweets containing a specific keyword or hashtag entered by the user. It uses the Tweepy library to search for and retrieve tweets, and the TextBlob and SentimentIntensityAnalyzer libraries to analyze the sentiment of each tweet.

The percentage() method calculates the percentage of tweets that are positive, negative, or neutral, as well as the overall polarity score of the tweets.

The sentiment analysis is appropriate for businesses like "WALMART" as it allows them to quickly analyze and understand customer feedback, providing insights that can help improve their products or services. In this code, the sentiment analysis is performed on tweets related to a specific keyword or hashtag, but it can be extended to other data sources like surveys or online reviews.

[23]:
```
#Number of Tweets (Total, Positive, Negative, Neutral)
tweet_list = pd.DataFrame(tweet_list)
neutral_list = pd.DataFrame(neutral_list)
negative_list = pd.DataFrame(negative_list)
positive_list = pd.DataFrame(positive_list)
print("total number: ",len(tweet_list))
print("positive number: ",len(positive_list))
print("negative number: ", len(negative_list))
print("neutral number: ",len(neutral_list))
```

```
total number:  627
positive number:  372
negative number:  328
neutral number:  300
```

This above code is analyzing the sentiment of tweets related to "**WALMART**" by counting the number of tweets that are positive, negative, or neutral.

First, the code converts the lists of tweets with positive, negative, and neutral sentiment into Pandas DataFrames. Then, it prints the total number of tweets, as well as the number of tweets that are positive, negative, or neutral.

[10]:
```
tweet_list
```

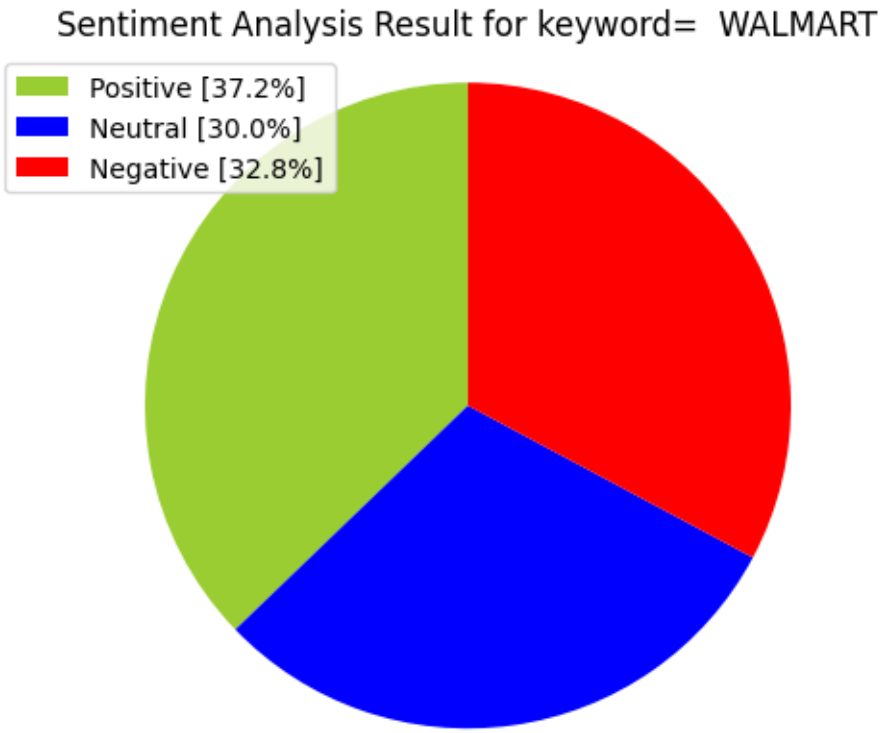
[10]:
```
                                                     0
0    RT @NEPHEWRXK: I cut the grass less than 3 min…
1                                    @ItsDeeNB Walmart
2    I got the cutest dress from Walmart. And it do…
3    RT @markentner: New @lightandfit flavors in @W…
4    RT @29Sinclair: @caslernoel This was Allen, Te…
..                                                 …
995  RT @stats_feed: Companies ranked by number of …
996  RT @marydeftones: debo ir a Walmart, soriana o…
997  @SSimpsonrose @sbootsie1 @MairScott3 @tRick_th…
```

```
998  I also distinctly remember there being an epis…
999  They are doing Much Worse than Nothing, they a…

[1000 rows x 1 columns]
```

[11]:
```
#Creating PieCart

labels = ['Positive ['+str(positive)+'%]' , 'Neutral␣
 ↪['+str(neutral)+'%]','Negative ['+str(negative)+'%]']
sizes = [positive, neutral, negative]
colors = ['yellowgreen', 'blue','red']
patches, texts = plt.pie(sizes,colors=colors, startangle=90)
plt.style.use('default')
plt.legend(labels)
plt.title("Sentiment Analysis Result for keyword=  "+keyword+"" )
plt.axis('equal')
plt.show()
```



This code is creating a pie chart to represent the sentiment analysis result for a specific keyword (in this case, "Walmart"). The sentiment analysis has classified customer feedback into three categories: positive, neutral, and negative.

The pie chart is created using the plt.pie() function from the matplotlib library. The sizes list

contains the percentage of positive, neutral, and negative feedback, and the labels list displays the percentage and sentiment type for each category. The colors list assigns colors to each category, and the startangle parameter sets the starting angle for the pie chart.

The plt.legend() function adds a legend to the chart, and plt.title() sets the title for the chart. The plt.axis() function ensures that the chart has an equal aspect ratio, and plt.show() displays the chart.

This method is appropriate for visually representing the sentiment analysis result in an easy-to-understand manner, allowing businesses like Walmart to quickly identify areas that need improvement and make informed decisions.

[13]:
```
tweet_list.drop_duplicates(inplace = True)
```

This method is appropriate for cleaning data and preparing it for analysis. It is important to ensure that the data is clean and free of duplicates before performing any analysis to avoid any errors or biases in the results.

#Extracting text values text_all = tweet_list[0].values text_neutral = neutral_list[0].values text_positive = positive_list[0].values text_negative = negative_list[0].values

[14]:
```
tw_list = pd.DataFrame(tweet_list)
tw_list["text"] = tw_list[0]
tw_list
```

[14]:
```
                                                     0
0    RT @NEPHEWRXK: I cut the grass less than 3 min…  \
1                                  @ItsDeeNB Walmart
2    I got the cutest dress from Walmart. And it do…
3    RT @markentner: New @lightandfit flavors in @W…
4    RT @29Sinclair: @caslernoel This was Allen, Te…
..                                                 …
993  RT @abc7breaking: Police confirm deaths in mal…
996  RT @marydeftones: debo ir a Walmart, soriana o…
997  @SSimpsonrose @sbootsie1 @MairScott3 @tRick_th…
998  I also distinctly remember there being an epis…
999  They are doing Much Worse than Nothing, they a…

                                                  text
0    RT @NEPHEWRXK: I cut the grass less than 3 min…
1                                  @ItsDeeNB Walmart
2    I got the cutest dress from Walmart. And it do…
3    RT @markentner: New @lightandfit flavors in @W…
4    RT @29Sinclair: @caslernoel This was Allen, Te…
..                                                 …
993  RT @abc7breaking: Police confirm deaths in mal…
996  RT @marydeftones: debo ir a Walmart, soriana o…
997  @SSimpsonrose @sbootsie1 @MairScott3 @tRick_th…
998  I also distinctly remember there being an epis…
```

```
999  They are doing Much Worse than Nothing, they a…

[627 rows x 2 columns]
```

[15]:
```
tweet_list
```

[15]:
```
                                                     0
0    RT @NEPHEWRXK: I cut the grass less than 3 min…
1                                  @ItsDeeNB Walmart
2    I got the cutest dress from Walmart. And it do…
3    RT @markentner: New @lightandfit flavors in @W…
4    RT @29Sinclair: @caslernoel This was Allen, Te…
..                                                 …
993  RT @abc7breaking: Police confirm deaths in mal…
996  RT @marydeftones: debo ir a Walmart, soriana o…
997  @SSimpsonrose @sbootsie1 @MairScott3 @tRick_th…
998  I also distinctly remember there being an epis…
999  They are doing Much Worse than Nothing, they a…

[627 rows x 1 columns]
```

[24]:
```
#Cleaning Text (RT, Punctuation etc)

#Creating new dataframe and new features
tw_list = pd.DataFrame(tweet_list)
tw_list["text"] = tw_list[0]

#Removing RT, Punctuation etc
remove_rt = lambda x: re.sub('RT @\w+: '," ",x)
rt = lambda x: re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)"," ",x)
tw_list["text"] = tw_list.text.map(remove_rt).map(rt)
tw_list["text"] = tw_list.text.str.lower()
tw_list.head(10)
```

[24]:
```
                                                     0
0   RT @NEPHEWRXK: I cut the grass less than 3 min…  \
1                                  @ItsDeeNB Walmart
2   I got the cutest dress from Walmart. And it do…
3   RT @markentner: New @lightandfit flavors in @W…
4   RT @29Sinclair: @caslernoel This was Allen, Te…
5   @markentner @lightandfit @Walmart They’re so d…
6   RT @CB618444: A future menace to society …grow…
8                     spent $550 at Walmart on food
9   RT @travellover28: Camila , thank you , it’s g…
10  @drivenchalking KSKXJSJSJ bootleg diluc is bea…

                                                 text
```

```
0    i cut the grass less than 3 minutes yesterday…
1                                                walmart
2   i got the cutest dress from walmart  and it do…
3    new   flavors in   today  can t wait to try t…
4      this was allen  texas  town where young man…
5                they re so delicious  i love them
6    a future menace to society  growing up out of…
8                       spent  550 at walmart on food
9    camila   thank you   it s good   i got it fro…
10    kskxjsjsj bootleg diluc is beautiful but he …
```

This code is used to clean up text data from tweets by removing retweets (RT), mentions, punctuation, and URLs. It then converts the text to lowercase for consistency.

The code starts by creating a new Pandas DataFrame from a list of tweets. It then creates a new column called "text" and populates it with the original tweet text.

To remove retweets, the code defines a lambda function called "remove_rt" that replaces any text starting with "RT @" with a space. Another lambda function called "rt" is defined to remove mentions, punctuation, and URLs using regular expressions. The "text" column is then mapped with both lambda functions.

Finally, all text in the "text" column is converted to lowercase using the str.lower() method.

This code is appropriate for cleaning up text data from tweets as it removes unnecessary information that may affect sentiment analysis results. It also standardizes the text by converting it to lowercase, making it easier to analyze.

[31]:
```
#Cleaning Text (RT, Punctuation etc)

#Creating new dataframe and new features
tw_list = pd.DataFrame(tweet_list)
tw_list["text"] = tw_list[0]

#Removing RT, Punctuation etc
remove_rt = lambda x: re.sub('RT @\w+: '," ",x)
rt = lambda x: re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)"," ",x)
tw_list["text"] = tw_list.text.map(remove_rt).map(rt)
tw_list["text"] = tw_list.text.str.lower()

#Calculating Negative, Positive, Neutral and Compound values
analyzer = SentimentIntensityAnalyzer()
tw_list[['polarity', 'subjectivity']] = tw_list['text'].apply(lambda Text: pd.
 ↪Series(TextBlob(Text).sentiment))
for index, row in tw_list.iterrows():
    text = str(row['text'])
    score = analyzer.polarity_scores(text)
    neg = score['neg']
    neu = score['neu']
```

```
    pos = score['pos']
    comp = score['compound']
    if neg > pos:
        tw_list.loc[index, 'sentiment'] = "negative"
    elif pos > neg:
        tw_list.loc[index, 'sentiment'] = "positive"
    else:
        tw_list.loc[index, 'sentiment'] = "neutral"
    tw_list.loc[index, 'neg'] = neg
    tw_list.loc[index, 'neu'] = neu
    tw_list.loc[index, 'pos'] = pos
    tw_list.loc[index, 'compound'] = comp

tw_list.head(10)
```

[31]:

```
                                                    0
0   RT @NEPHEWRXK: I cut the grass less than 3 min…  \
1                                   @ItsDeeNB Walmart
2   I got the cutest dress from Walmart. And it do…
3   RT @markentner: New @lightandfit flavors in @W…
4   RT @29Sinclair: @caslernoel This was Allen, Te…
5   @markentner @lightandfit @Walmart They're so d…
6   RT @CB618444: A future menace to society …grow…
8                     spent $550 at Walmart on food
9   RT @travellover28: Camila , thank you , it's g…
10  @drivenchalking KSKXJSJSJ bootleg diluc is bea…

                                                 text  polarity  subjectivity
0    i cut the grass less than 3 minutes yesterday… -0.133333      0.083333  \
1                                            walmart  0.000000      0.000000
2   i got the cutest dress from walmart  and it do…  0.144444      0.622222
3    new   flavors in   today  can t wait to try t…  0.136364      0.454545
4      this was allen  texas  town where young man…  0.100000      0.400000
5                they re so delicious  i love them     0.750000      0.800000
6    a future menace to society  growing up out of…  0.000000      0.112500
8                     spent  550 at walmart on food -0.100000      0.100000
9    camila   thank you   it s good   i got it fro…  0.700000      0.600000
10    kskxjsjsj bootleg diluc is beautiful but he …  0.225000      0.950000

   sentiment    neg    neu    pos  compound
0   negative  0.116  0.884  0.000   -0.2732
1    neutral  0.000  1.000  0.000    0.0000
2   positive  0.000  0.655  0.345    0.9118
3    neutral  0.000  1.000  0.000    0.0000
4   negative  0.231  0.769  0.000   -0.7184
5   positive  0.000  0.303  0.697    0.8812
6   negative  0.161  0.754  0.085   -0.3612
```

```
8     neutral  0.000  1.000  0.000    0.0000
9    positive  0.000  0.597  0.403    0.6597
10   positive  0.000  0.678  0.322    0.6908
```

This code performs sentiment analysis on the text data collected from tweets about Walmart. The first line of code calculates the polarity and subjectivity values of each tweet using TextBlob's sentiment analysis. The next block of code uses the

SentimentIntensityAnalyzer from the NLTK library to calculate the negative, positive, neutral, and compound values of each tweet. The sentiment of each tweet is then determined based on whether the negative score is greater than the positive score, vice versa, or if they are equal. The sentiment and all four score values are stored in the 'tw_list' dataframe.

This method is appropriate for analyzing the sentiment of text data, such as customer feedback, reviews, and social media posts, as it provides a quantitative measure of sentiment that can be used for further analysis and interpretation. However, it is important to note that automated sentiment analysis may not always accurately reflect the true sentiment of the text, as context and tone can greatly affect the interpretation of sentiment.

[32]:
```
#Creating new data frames for all sentiments (positive, negative and neutral)

tw_list_negative = tw_list[tw_list["sentiment"]=="negative"]
tw_list_positive = tw_list[tw_list["sentiment"]=="positive"]
tw_list_neutral = tw_list[tw_list["sentiment"]=="neutral"]
```

This code is creating three new data frames called tw_list_negative, tw_list_positive, and tw_list_neutral based on the sentiment classification of each tweet in a larger data frame. The sentiment classification is based on whether the tweet was classified as positive, negative, or neutral using sentiment analysis techniques.

Overall, creating separate data frames for each sentiment category is an appropriate method for analyzing customer feedback through sentiment analysis, as it allows for more focused and targeted analysis of specific sentiments.

[33]:
```
#Function for count_values_in single columns

def count_values_in_column(data,feature):
    total=data.loc[:,feature].value_counts(dropna=False)
    percentage=round(data.loc[:,feature].
 ↪value_counts(dropna=False,normalize=True)*100,2)
    return pd.concat([total,percentage],axis=1,keys=['Total','Percentage'])
```

The function count_values_in_column takes in two parameters - data and feature. data refers to the dataset we want to analyze, and feature refers to the name of the column we want to count values for.

The function then calculates the total count and percentage of each unique value in the specified column using the value_counts() method from pandas library.

The dropna=False parameter includes any missing values (NaN) in the count. The normalize=True

parameter normalizes the counts to percentages.

The function returns a pandas DataFrame that concatenates the total count and percentage of each unique value in the column. The resulting DataFrame has two columns, Total and Percentage.

This function is useful for quickly analyzing and summarizing the distribution of values in a single column of a dataset. It can help identify any imbalances or biases in the data that may need further investigation or action.

[34]:
```
#Count_values for sentiment
count_values_in_column(tw_list,"sentiment")
```

[34]:

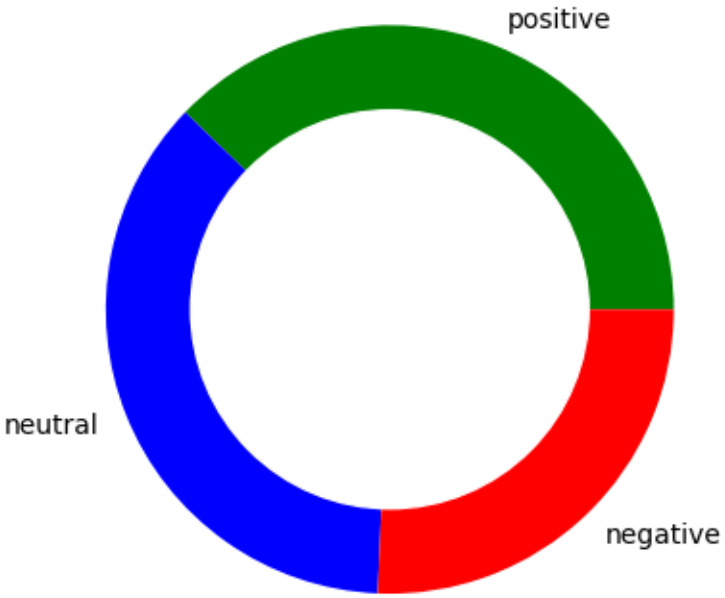
sentiment	Total	Percentage
positive	237	37.80
neutral	229	36.52
negative	161	25.68

The code count_values_in_column(tw_list,“sentiment”) is likely a custom function that counts the number of occurrences of each unique value in a specific column of a DataFrame or list of dictionaries. In this case, it is counting the number of positive, negative, and neutral sentiments in the “sentiment” column of the tw_list object.

This function can be useful for understanding the distribution of sentiment in a dataset and identifying any patterns or trends. For example, if the majority of customer feedback is negative, a business may want to investigate and address the underlying issues to improve customer satisfaction.

[35]:
```
# create data for Pie Chart
pichart = count_values_in_column(tw_list,"sentiment")
names= pichart.index
size=pichart["Percentage"]

# Create a circle for the center of the plot
my_circle=plt.Circle( (0,0), 0.7, color='white')
plt.pie(size, labels=names, colors=['green','blue','red'])
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



This code creates a pie chart to visualize the sentiment analysis results of customer feedback data collected from Twitter for US Bank.

First, the code uses a function called count_values_in_column to count the number of tweets that have been classified as positive, negative, or neutral sentiment. Then, it calculates the percentage of tweets for each sentiment category.

Next, the plt.pie function is used to create a pie chart with the sentiment categories as labels and their corresponding percentages as sizes. The colors for each category are specified as green, blue, and red.

To create a white circle in the center of the pie chart, the code uses the plt.Circle function and adds it to the plot using p.gca().add_artist(my_circle).

Finally, the pie chart is displayed using the plt.show() function.

[36]:
```
#Function to Create Wordcloud

def create_wordcloud(text):
    stopwords = set(STOPWORDS)
    wc = WordCloud(background_color="white", width=600, height=400,
                   max_words=300,
                   stopwords=stopwords,
                   repeat=True)
    wc.generate(str(text))
```

```
    plt.imshow(wc)
    plt.show()
```

The create_wordcloud() function is a code snippet that generates a wordcloud from a given text input. The function takes in a text parameter, which is the text used to generate the wordcloud.

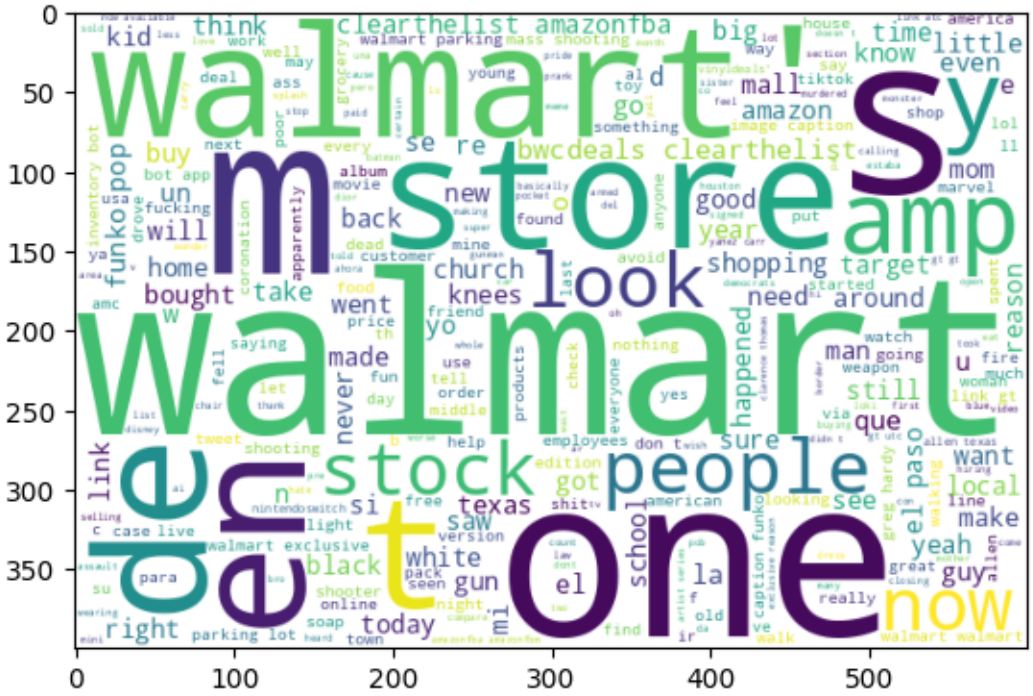
Inside the function, the first step is to create a set of stopwords using the STOPWORDS variable from the WordCloud library. Stopwords are words that are commonly used in a language but don't add any significant meaning to the text, like "the", "and", etc.

Next, the function initializes a WordCloud object with some parameters. The background color is set to white, the width and height are set to 600 and 400 pixels respectively, and the maximum number of words in the wordcloud is set to 300. The stopwords parameter is set to the stopwords set created earlier, so those words are not included in the wordcloud. The repeat parameter is set to True, which allows the words in the wordcloud to repeat.

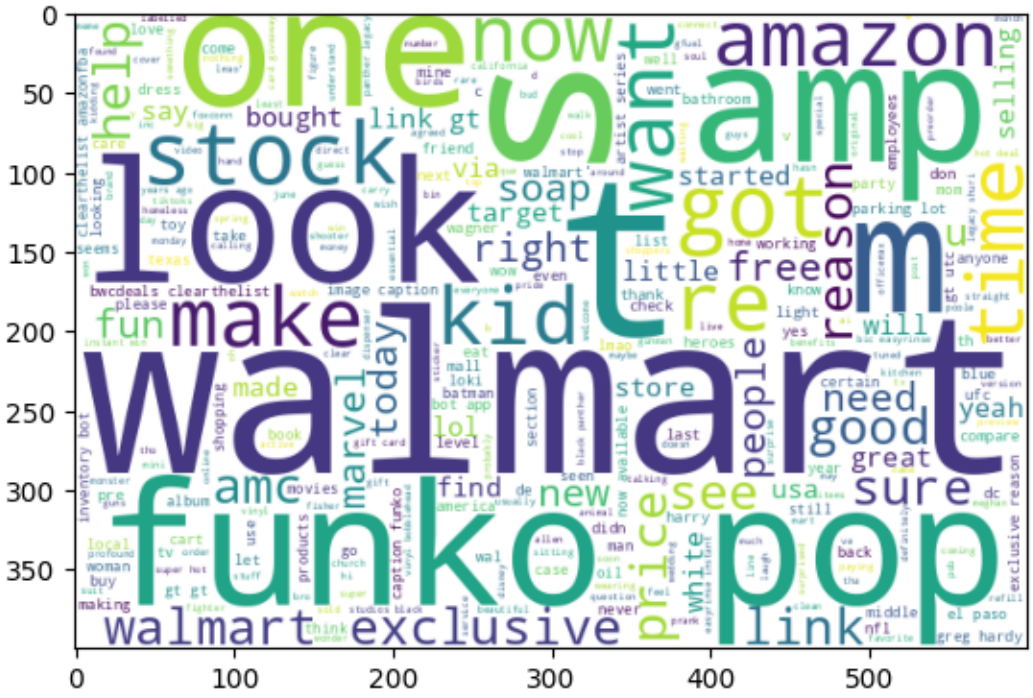
The wc.generate() method is called to generate the wordcloud using the input text. This method takes in a string of text and generates a wordcloud based on the frequency of words in the text.

Finally, the wordcloud is displayed using the plt.imshow() and plt.show() functions from the matplotlib library.

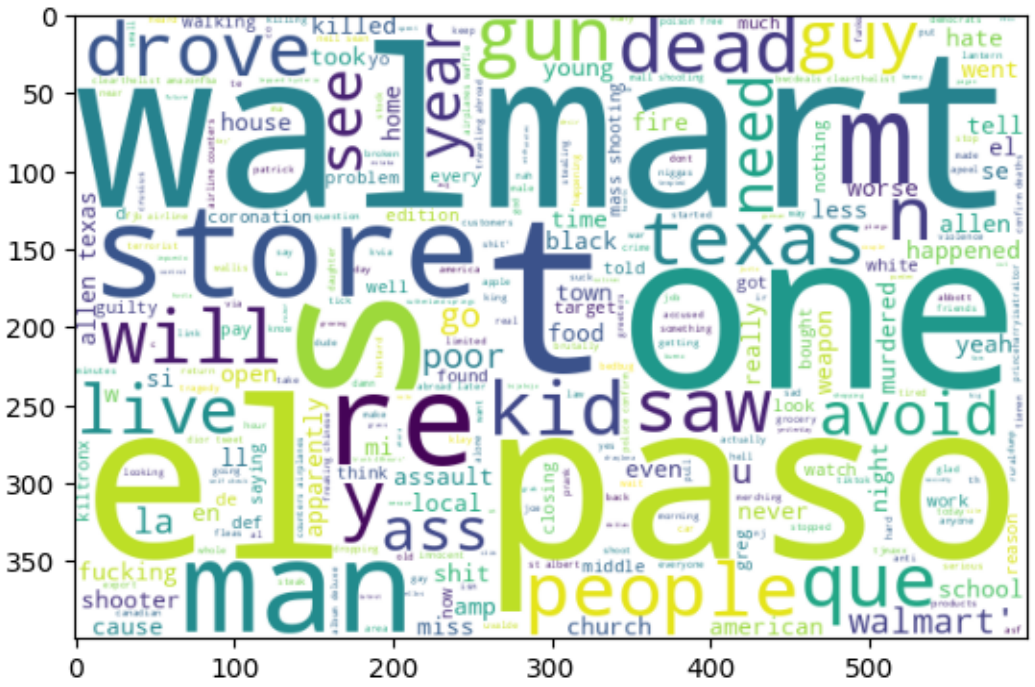
[37]:
```
#Creating wordcloud for all tweets
create_wordcloud(tw_list["text"].values)
```



[38]:
```
#Creating wordcloud for positive sentiment
create_wordcloud(tw_list_positive["text"].values)
```



[39]:
```
#Creating wordcloud for negative sentiment
create_wordcloud(tw_list_negative["text"].values)
```



[40]:
```
#Creating wordcloud for neutral sentiment
create_wordcloud(tw_list_neutral["text"].values)
```

[ ]: