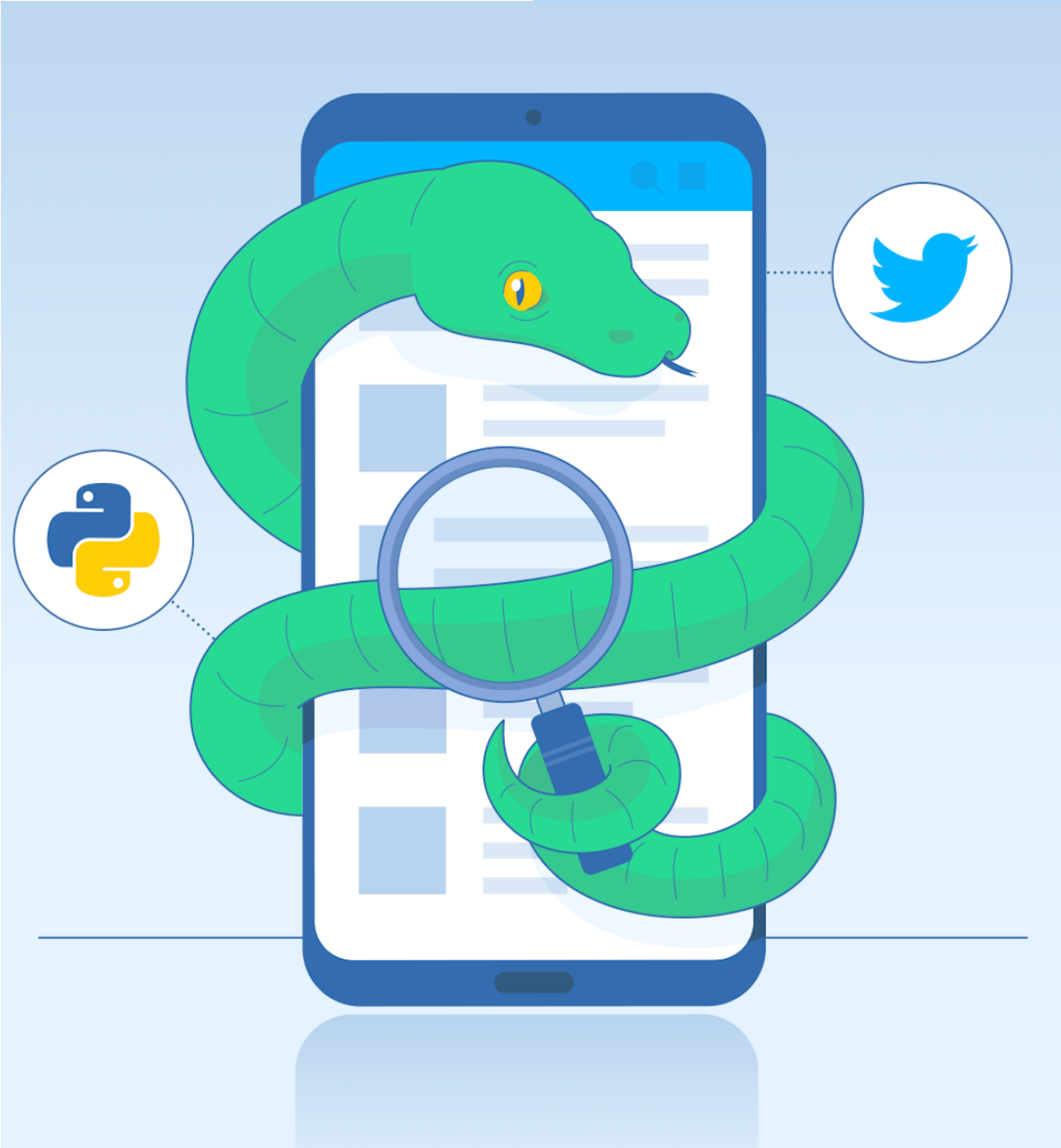


SENTIMENT ANALYSIS USING TEXT MINING

Project Report

Team Cook

Hélène LI, Kai NAKAMURA, Emeline Yisu XU

SUMMARY

INTRODUCTION

Sentiment analysis refers to the process of determining the sentiment that is contained in a piece of writing, thereby using text mining technique. Simply put, it analyses if the text renders a positive, negative or neutral message. Sentiment analysis is also know as opinion mining, deriving the opinion or attitude of a speaker.

Sentiment analysis is therefore extremely useful as it serves many purposes, ranging from business, politics to public actions. For example, using python programming and more specifically sentiment analysis, marketing companies are able to identify customers's feelings and perceptions regarding their products/brands. It enables them to understand how their customers respond to their campaigns, product launches, and functionality preferences. In this sense, sentiment analysis efficiently addresses problems that can arise from potential wrong customer behavior analysis. Moreover, sentiment analysis can also be used in the political field, to the extent that it allows to keep track of political views, detect consistency and inconsistency between statements and actions at the government level. In regards of public actions, sentiment analysis can be used to analyse social phenomena, thereby spotting potentially dangerous situations and determining the general mood of the society.

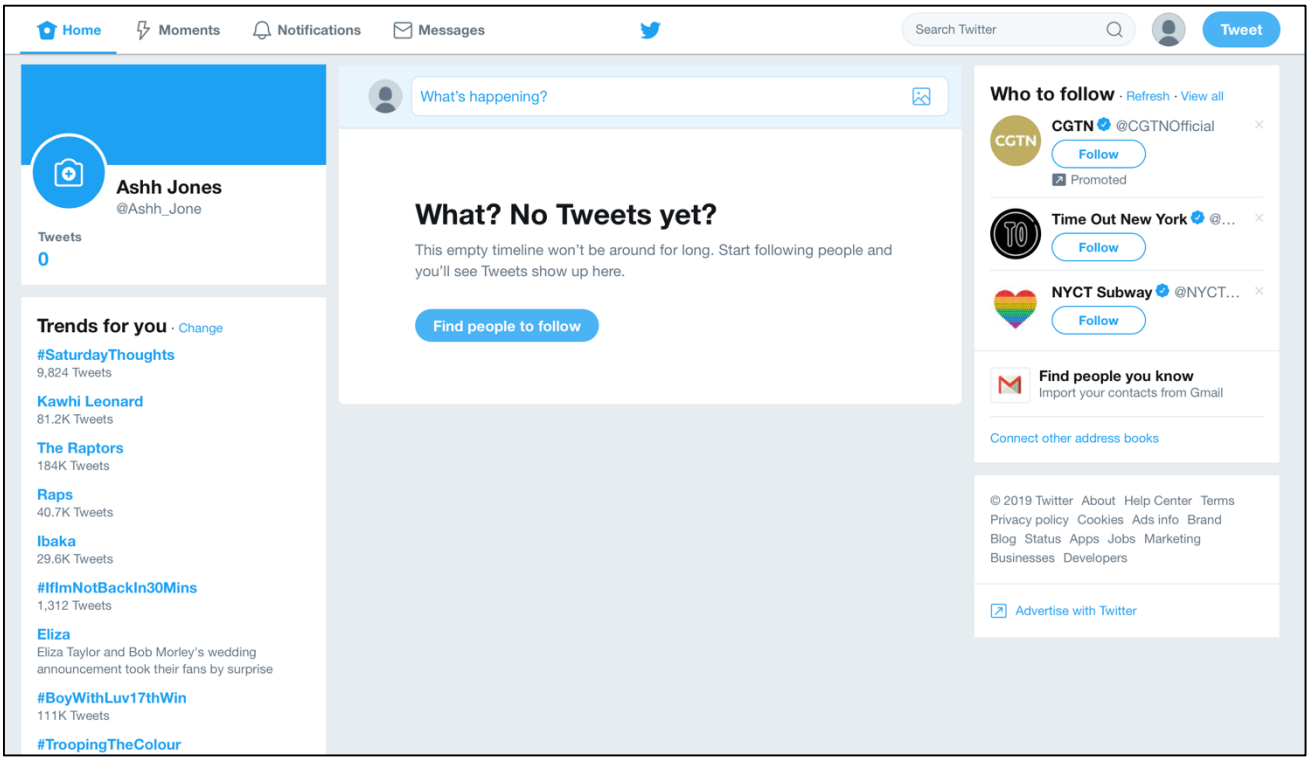
Therefore, we can assert that in this digital era that we are living, the age of getting meaningful insights from social media data has now arrived. These insights are important as they are strongly linked to consumer behaviour: what our customers want, what are customers like and dislike about our products, what their buying signals are, what their decision process looks like and so on…Indeed, as more and more content is created and shared online, through various social channels, blogs, review sites and so on, we are becoming more and more vocal and open about our experiences online. According to a study carried out by Zendesk, 45% of people share bad customer service experiences and 30% share good customer service experiences via social media. This phenomenon therefore highlights the need and desire for businesses to mine this information to gain precious business insights.

That is why sentiment analysis can help a better understanding of social sentiments. In this study, we used the python program by taking the example of Twitter. The program will extract tweets and analyse the sentiments that are underlying. In this way, we will be able to unlock the hidden value of text in order to understand people's opinions and needs in order to make better, more informed, business decisions.

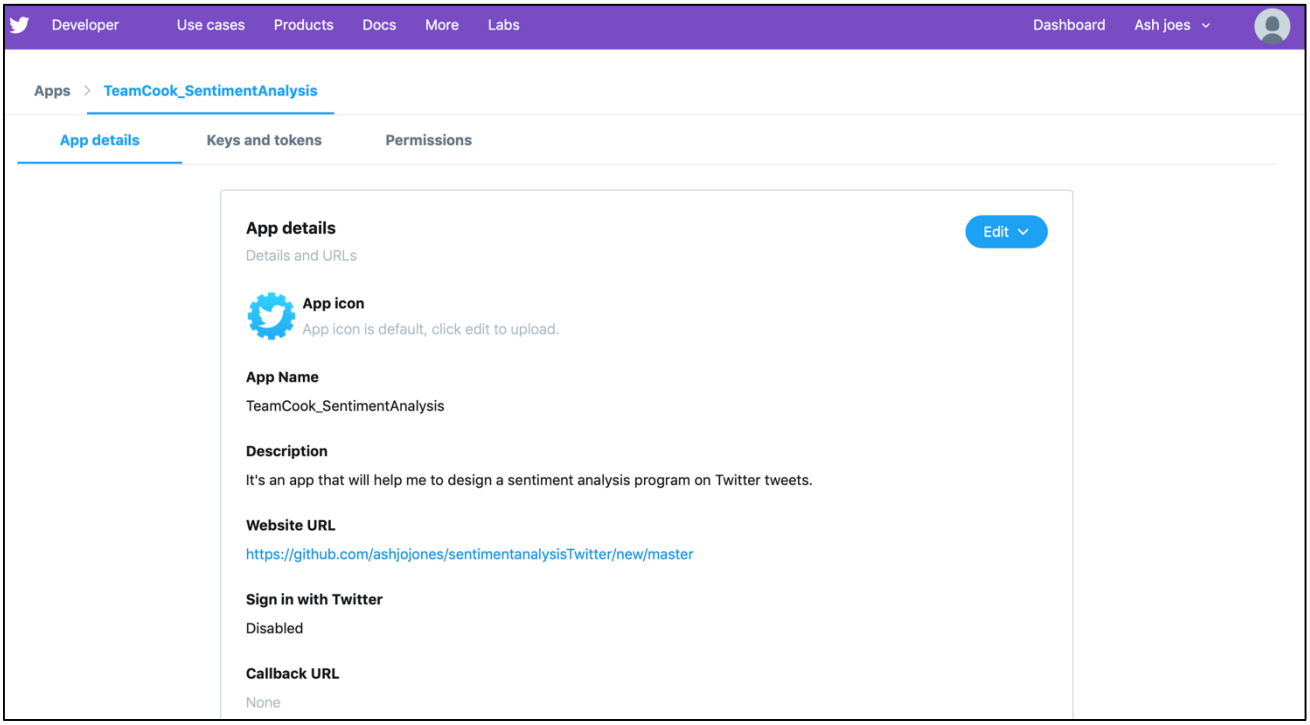
I. PREREQUISITES

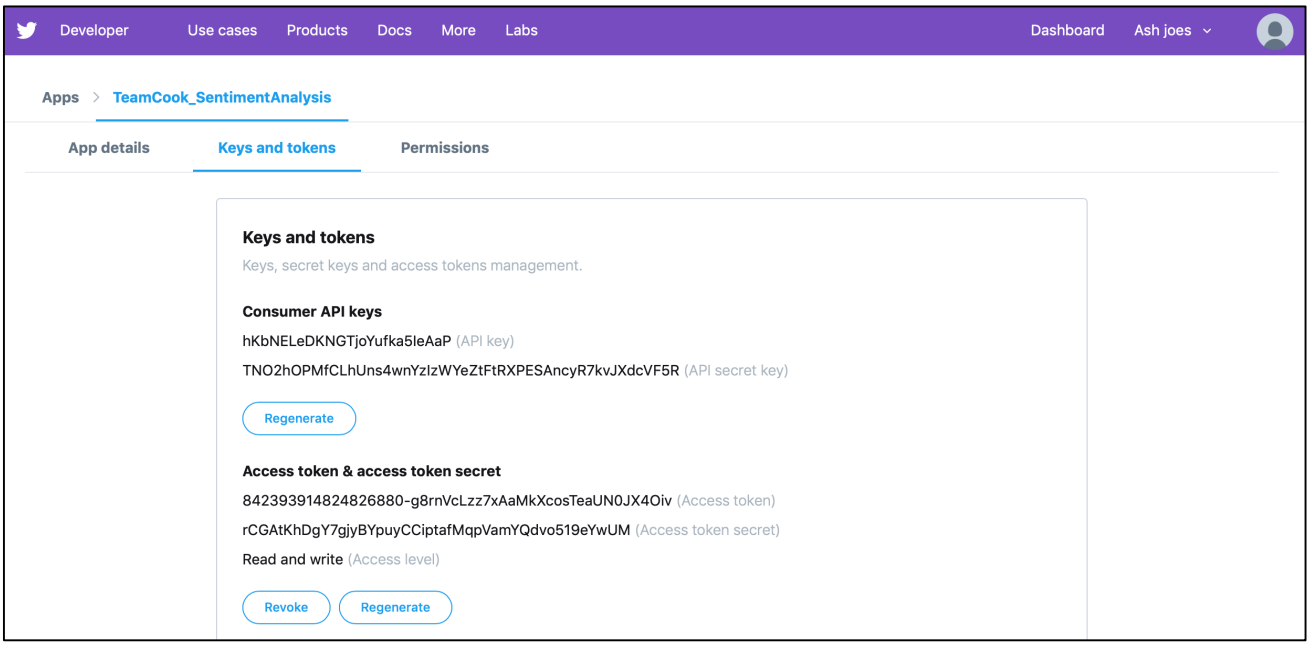
1. Twitter and Twitter API

First, it is necessary to register a **Twitter** account on https://twitter.com/ if you do not have one. So, we created one whether it is a new one or your personal one, it works either way.



Then, we have to set up a **Twitter API** (Application Programming Interfaces) on https://apps.twitter.com/ . This platform provides three tiers for searching real time Tweets. You have to log in with the Twitter account you just created and then, create an 'Apps'. Therefore, you will have access to "**Keys and Access tokens**" or credentials, which allow you to use those in your Python code to access Twitter data.





Once everything is set up, you can create function to download tweets based on a search keyword.

2. Libraries

Before moving on to the core of the code, you have to install a few libraries, as in:

- **Tweepy**: it is the python client for Twitter API.
- **TextBlob**: it processes textual data, known to be used for sentiment analysis.

- **Numpy**: it provides fast precompiled functions for mathematical and numerical routines.
- **Matplotlib**: it provides both a very quick way to visualize data from Python and publication-quality figures in many formats.
- **Pandas**: it carries out entire data analysis workflow in Python without having to switch to a more domain specific language.
- **Re**: it specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression.

In order to install those, you have to go through the terminal of your computer by using the command "pip install tweepy".

```
from tweepy import API
from tweepy import Cursor
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

from textblob import TextBlob

import twitter_credentials

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
```

3. Twitter Credentials

The twitter credentials is a file called "twitter_credentials.py" where you will use the keys and tokens to access the Twitter API account mentioned above.
Therefore, in the second file, called "sentiment_analysis_twitter_data.py", you can import "twitter credentials" to gain access. It is the main file where we will access the Twitter API data, authenticate the access, process live tweets, process a basic listener, analyse and categorize tweets and analyse the sentiment of the tweets based on the keyword chosen.

```
# Variables that contains the user credentials to access Twitter API
ACCESS_TOKEN = "842393914824826880-g8rnVcLzz7xAaMkXcosTeaUN0JX4Oiv"
ACCESS_TOKEN_SECRET = "rCGAtKhDgY7gjyBYpuyCCiptafMqpVamYQdvo519eYwUM"
CONSUMER_KEY = "hKbNELeDKNGTjoYufka5IeAaP"
CONSUMER_SECRET = "TNO2hOPMfCLhUns4wnYzIzWYeZtFtRXPESAncyR7kvJXdcVF5R"
```

II. TWEEPY

Tweepy is the python client for Twitter API, thanks to that library you can retrieve data from Twitter. A wide range of function used in the project comes from its website specific to Tweepy https://tweepy.readthedocs.io/en/latest/getting_started.html .

1. Authorize Twitter API client

First of all, to begin the process we need to register our client application with Twitter we create a TwitterClient class. This class contains all the methods to interact with Twitter API and parsing tweets. We use __init__ function to handle the authentication of API client.

```
class TwitterClient():
    def __init__(self, twitter_user=None):
        self.auth = TwitterAuthenticator().authenticate_twitter_app()
        self.twitter_client = API(self.auth)

        self.twitter_user = twitter_user

    def get_twitter_client_api(self):
        return self.twitter_client

    def get_user_timeline_tweets(self, num_tweets):
        tweets = []
        for tweet in Cursor(self.twitter_client.user_timeline, id=self.twitter_user).items(num_tweets):
            tweets.append(tweet)
        return tweets

    def get_friend_list(self, num_friends):
        friend_list = []
        for friend in Cursor(self.twitter_client.friends, id=self.twitter_user).items(num_friends):
            friend_list.append(friend)
        return friend_list

    def get_home_timeline_tweets(self, num_tweets):
        home_timeline_tweets = []
        for tweet in Cursor(self.twitter_client.home_timeline, id=self.twitter_user).items(num_tweets):
            home_timeline_tweets.append(tweet)
        return home_timeline_tweets
```

Indeed, class TwitterClient, **__init__** function is made to authenticate through the Twitter API. Then, **get_user_timeline_tweets** function will allow you to determine how many tweets you want to extract. Similarly**, get_friend_list** function, like the previous one, allow you to see the tweets of your friends. Also, **get_home_timeline_tweets** function, the homepage timeline is all the tweets of people's you are following, so you would be able to extract their tweets and use it. Those 3 last functions use a loop through "cursor" to retrieve tweets data of your Twitter account. For instance, "self" means your account and, "num_tweets" specifies the number of tweets to retrieve from the home page.

Unlike basic **auth**, we must do the **OAuth** before we can start using the API, meaning that we must complete the following steps:

1. Get a request token from Twitter
2. Redirect user to twitter.com to authorize our application
3. If using a callback, twitter will redirect the user to us. Otherwise the user must manually supply us with the verifier code.
4. Exchange the authorized request token for an access token.

```
class TwitterAuthenticator():

    def authenticate_twitter_app(self):
        auth = OAuthHandler(twitter_credentials.CONSUMER_KEY, twitter_credentials.CONSUMER_SECRET)
        auth.set_access_token(twitter_credentials.ACCESS_TOKEN, twitter_credentials.ACCESS_TOKEN_SECRET)
        return auth
```

In this class TwitterAuthenticator, by creating the OAutHandler, it takes the argument in the "twitter_credentials.py" called consumer key, consumer secret, access token and access token secret.

2. Twitter Streamer

Twitter Streamer is a class for streaming and processing live tweets. Then, stream_tweets function handles Twitter authentication and its connection to Twitter streaming API. Stream.filter () allow us to capture data by using keyword through tweets and hashtags.

```
class TwitterStreamer():

    def __init__(self):
        self.twitter_autenticator = TwitterAuthenticator()

    def stream_tweets(self, fetched_tweets_filename, hash_tag_list):
        # This handles Twitter authentification & connection to Twitter Streaming API
        listener = TwitterListener(fetched_tweets_filename)
        auth = self.twitter_autenticator.authenticate_twitter_app()
        stream = Stream(auth, listener)

        # This line filter Twitter Streams to capture data by the keywords:
        stream.filter(track=hash_tag_list)
```

3. Twitter Listener

One of the main usage cases of tweepy is monitoring for tweets and doing actions when some event happens. The key component of that is the StreamListener object, which monitors tweets in real time and catches them. It is a basic listener which handles tweets that are received from the stream. Thus, prints received tweets to stdout which handles data received from the stream. StreamListener has several methods:

- on_data() is activated whenever the tweet has been heard. This method of a **stream**listener receives all messages and calls functions according to the message type
- on_error() is used to serve as a error handler for our listener. Indeed, Error 420 are sent to our listener because of Twitter's rate limit policy and whenever it happends, it will prompt our listener to disconnect.

```
class TwitterListener(StreamListener):

    def __init__(self, fetched_tweets_filename):
        self.fetched_tweets_filename = fetched_tweets_filename

    def on_data(self, data):
        try:
            print(data)
            with open(self.fetched_tweets_filename, 'a') as tf:
                tf.write(data)
            return True
        except BaseException as e:
            print("Error on_data %s" % str(e))
        return True

    def on_error(self, status):
        if status == 420:
            # Returning False on_data method in case rate limit occurs.
            return False
        print(status)
```

III. DATA ANALYSIS

This part of the code is focused on the data analysis of the tweets, basically it analyses and categorizes content from tweets.

First, we call clean_tweet() method to remove links and special characters from the tweet using simple regex. Tweets contain many slang words and punctuation marks. We need to clean our tweets before they can be used for training the machine learning model.
Once it is cleaned, we can move to the sentiment analysis. Textblob is a high level library built over the top of NLTK library. So, we create a Texblob object, this library will process:

- Split words from the body text meaning tokenize the tweet.
- Remove stopwords from the tokens meaning removing irrevelant words as "I" or "you".
- Do POS (Part Of Speech) which is selecting tagging of the tokens and significant tokens/features (i.e. adjectives, adverbs…).
- Pass the tokens to a sentiment classifier which classifies the tweet sentiment as positive, negative or neutral by assigning it to a polarity: 1, 0, -1.

The sentiment classifier is created through Texblob, it uses a movie review dataset in which reviews has been labelled as positive or negative. Those features are extracted from positive and negative reviews. The training data is trained on a Naïve Bayes Classifier where data are retrieved and labelled as a positive feature or negative feature.

```
class TweetAnalyzer():

    def clean_tweet(self, tweet):
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

    def analyze_sentiment(self, tweet):
        analysis = TextBlob(self.clean_tweet(tweet))

        if analysis.sentiment.polarity > 0:
            return 1
        elif analysis.sentiment.polarity == 0:
            return 0
        else:
            return -1
```

Each tweet is specific, it has an ID number, a content, a length, a date, a number of likes and retweets. Data_frame() function will help with the organization or those tweets, the tweets will be organized in a table according to "ID", "Date" etc.

```
    def tweets_to_data_frame(self, tweets):
        df = pd.DataFrame(data=[tweet.text for tweet in tweets], columns=['tweets'])

        df['id'] = np.array([tweet.id for tweet in tweets])
        df['len'] = np.array([len(tweet.text) for tweet in tweets])
        df['date'] = np.array([tweet.created_at for tweet in tweets])
        df['source'] = np.array([tweet.source for tweet in tweets])
        df['likes'] = np.array([tweet.favorite_count for tweet in tweets])
        df['retweets'] = np.array([tweet.retweet_count for tweet in tweets])

        return df
```

The function __main__ will help us to get the access to the API and interact with the twitter_client and tweet_analyzer as mentioned previously. Then user_timeline is a built-in function of tweepy twitter client in which we can specify the keyword filter = “fifawwc” for example, and the number of tweets processing number = 200. Then, we added another column to the previous data_frame() to store the sentiment of the tweet.

```
if __name__ == '__main__':

    twitter_client = TwitterClient()
    tweet_analyzer = TweetAnalyzer()

    api = twitter_client.get_twitter_client_api()
    filter = "fifawwc"
    number = 200
    tweets = api.user_timeline(screen_name=filter, count=number)

    df = tweet_analyzer.tweets_to_data_frame(tweets)
    df['sentiment'] = np.array([tweet_analyzer.analyze_sentiment(tweet) for tweet in df['tweets']])
```

Therfore, the output in the terminal is:

tweets	id	len	date	source	likes	retweets	sentiment
Heading back to D.C. Many great things are hap...	1137052508231069696	69	2019-06-07 17:43:38	Twitter for iPhone	24954	5369	1
For all of the money we are spending, NASA sho...	1137051097955102720	140	2019-06-07 17:38:01	Twitter for iPhone	34567	9078	0
Democrats are incapable of doing a good and so...	1137048453173862400	67	2019-06-07 17:27:31	Twitter for iPhone	26769	6692	1
If we are able to make the deal with Mexico, &...	1137045728344170496	144	2019-06-07 17:16:41	Twitter for iPhone	28860	7695	1
...and have no intention of doing anything oth...	1137040975258198017	140	2019-06-07 16:57:48	Twitter for iPhone	41943	9993	-1
Nervous Nancy Pelosi is a disgrace to herself ...	1137040971311353856	140	2019-06-07 16:57:47	Twitter for iPhone	51047	13153	-1
China is subsidizing its product in order that...	1137035722043547649	139	2019-06-07 16:36:55	Twitter for iPhone	31011	8045	1
HAPPY BIRTHDAY to our great @VP Mike Pence! ht...	1137007514166255621	67	2019-06-07 14:44:50	Twitter for iPhone	51017	9458	1
https://t.co/xrrzYhPu0i	1136958356101505024	23	2019-06-07 11:29:30	Twitter for iPhone	34167	10969	0
https://t.co/lNh7hQiQTN	1136951503799881728	23	2019-06-07 11:02:16	Twitter for iPhone	35769	9499	0

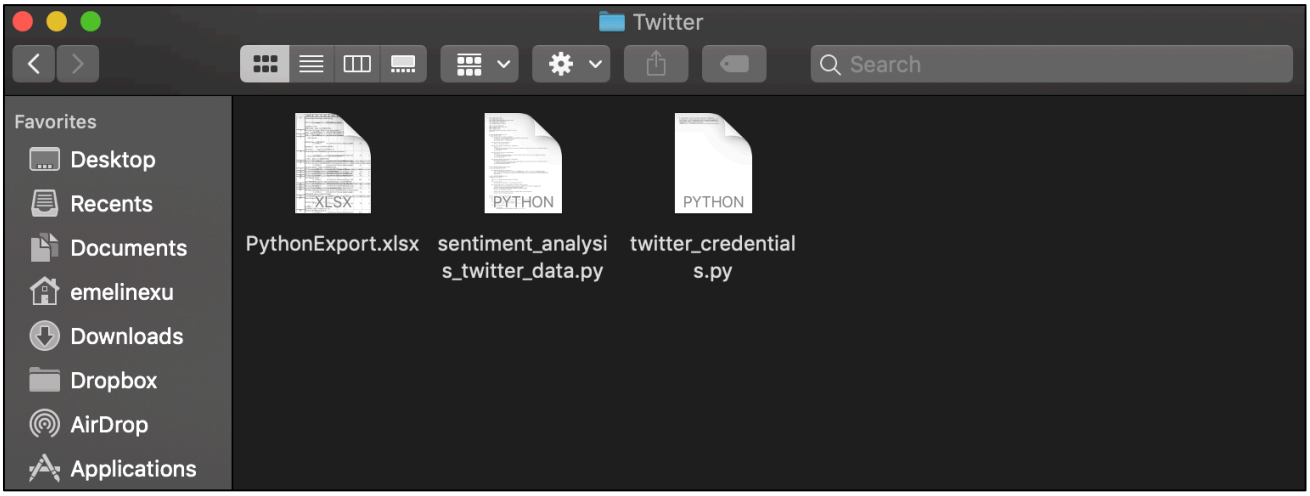
IV. DATA VISUALIZATION

1. Table in Excel

We found it easier to have an Excel table for the output table in the terminal. Thus, we created another file named "PythonExport.xlsx" which will transform the dataframe df() to excel and save it in the current folder with "twitter_credentials.py" and "sentiment_analysis_twitter_data.py".

```
# Export to excel
writer = ExcelWriter('PythonExport.xlsx') # Create file
df.to_excel(writer,'Sheet1') # Transform dataframe df to excel sheet named "sheet1"
writer.save() # Save into current folder

print(df.head(10))
```



The Excel table in PythonExport.xlsx will then look like:

	tweets	id	len	date	source	likes	retweets	sentiment
0	Gooooooooood morning, football fans Who is rea	1.13759E+18	139	2019-06-09 05:20:00	Twitter Ads Composer	266	77	1
1	First game. First goals. Highs. Lows. And emotions. #F	1.13758E+18	124	2019-06-09 04:45:00	Twitter Media Studio	91	21	1
2	#BRA's @MonicaHickmannA says there are no favouri	1.13752E+18	140	2019-06-09 00:30:00	Twitter Media Studio	311	61	0
3	✨ Giulia Gwinn ✨ @DFB_Frauen | #FIFAWWC High	1.13751E+18	118	2019-06-08 23:45:00	Twitter Media Studio	339	82	0
4	On the menu with @TheMatildas goal machine @sam	1.13748E+18	139	2019-06-08 22:01:15	Twitter Web Client	189	32	1
5	Tonight's #PlayeroftheMatch presented by @Visa for	1.13748E+18	140	2019-06-08 21:59:17	Twitter Media Studio	222	43	1
6	The first-ever goal at the #FIFAWWC for #BanyanaBan	1.13748E+18	140	2019-06-08 21:57:00	Twitter Media Studio	346	98	1
7	RT @Janinevanwyk5: Wow what a game full of emotic	1.13748E+18	139	2019-06-08 21:56:41	Twitter for iPhone	0	332	1
8	@ashlnhrris We hear ya. https://t.co/14TyiWlSQg	1.13748E+18	47	2019-06-08 21:53:43	Twitter for iPhone	109	8	0
9	We got some photos did. #USA #FIFAWWC https://t.c	1.13748E+18	61	2019-06-08 21:51:42	Twitter for iPhone	1063	124	0
10	@ashlynkriegers yeah we can do that for you. https://	1.13748E+18	68	2019-06-08 21:43:40	Twitter Web Client	119	28	0
11	Alex Morgan, #USA 2019 #FIFAWWC https://t.co/Knv	1.13747E+18	81	2019-06-08 21:41:30	Twitter for iPhone	840	148	0
12	GUESS WHICH TEAM HAD THEIR #FIFAWWC SQUAD P	1.13747E+18	79	2019-06-08 21:36:11	Twitter Web Client	641	52	0
13	Ok, cool. Two secs.	1.13747E+18	19	2019-06-08 21:35:12	Twitter Web Client	318	9	1

2. Pie chart of the sentiment analysis

Nevertheless, it is always better to have a visual of the data found. Therefore, we created a loop that combine positive, negative and neutral tweets in a pie chart.

Index is a built-in function in Pandas data frames, meaning that its values will be used as row labels or column "name".

Sentimentpercentage is a built-in function in Textblob that calculate the percentage of positive, negative and neutral tweets.

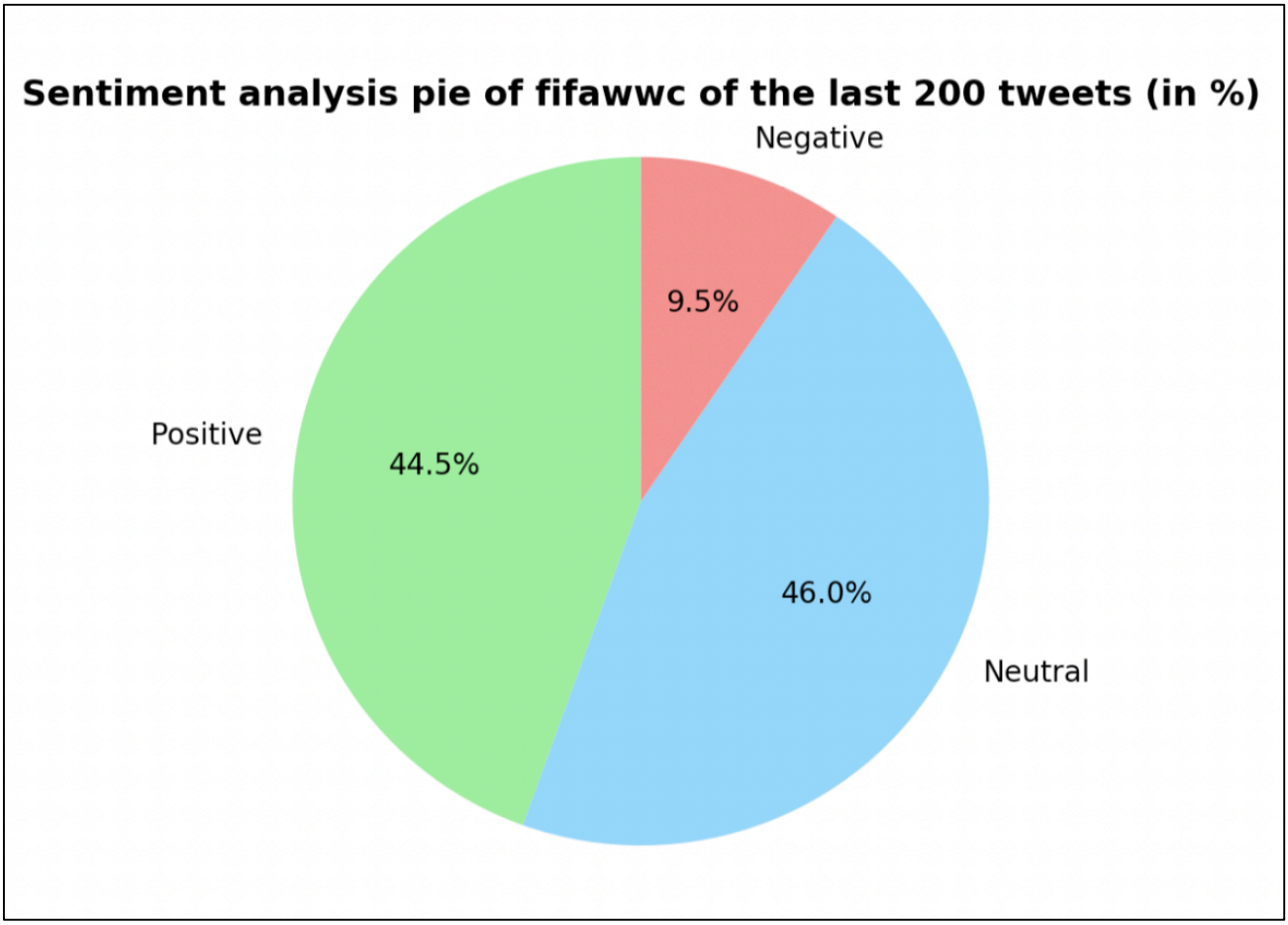
Colors, figureObject, pieLabels are layout of the pie chart.

Plt.title is the title we will be given to the pie chart using filter which is the keyword we previously used and, the str(number) which is the number of tweets we were analysing.

```
    Positive = 0
    Neutral = 0
    Negative = 0
    for index in range(len(df['sentiment'])):
        if df['sentiment'][index] == 1:
            Positive = Positive + 1
        if df['sentiment'][index] == 0:
            Neutral = Neutral + 1
        if df['sentiment'][index] == -1:
            Negative = Negative + 1

    pieLabels = 'Positive', 'Neutral', 'Negative'
    sentimentPercentage = [Positive, Neutral, Negative]
    colors = ['lightgreen', 'lightskyblue', 'lightcoral']
    figureObject, axesObject = plt.subplots()
    axesObject.pie(sentimentPercentage,labels=pieLabels,colors=colors,autopct='%1.1f%%',startangle=90)
    axesObject.axis('equal')
    plt.title("Sentiment analysis pie of " + filter + " of the last " + str(number) + " tweets (in %)",
fontweight="bold")
    plt.show()
```

Thus, another window will pop up in our computer screen after running the code in the terminal and, show the pie chart below:


Sentiment analysis pie of fifawwc of the last 200 tweets (in %)
Negative
9.5%
Positive
44.5%
46.0%
Neutral

V. ANALYSIS OF RESULT

As we took the keyword **FifaWWC**, the Excel table will help us know how many tweets has been liked or retweets. It gives us insights on the event, whether people liked it or not. If most of them has a positive sentiment, thanks to the textmining and data analysis, we can conclude that FifaWWC is likely appreciated by Twitter users. You can organize tweets that were the most liked and retweets with Excel solver. Hence, you will find the Twitter user that is influent and, can be a good sponsor to the next FifaWWC.

However, if many remain neutral, as described in the pie chart, companies can try to find out how they can better mediatize and reach out more people. Perhaps, it has not been advertised enough, thus the large percentage of Neutral. For instance, companies can target Twitter users that has the most retweeted and likes on their tweet, meaning that this user is a potential user who could better advertise the FifaWWC.

CONCLUSION

Emotions are essential to effective communication between humans, so if we want machines to handle texts in the same way, we need teach them how to detect emotions and classify text as positive, negative or neutral. That's where sentiment analysis comes into play. It's the automated process of understanding an opinion about a given subject from written or spoken language.

Every minute of the day, 156 million emails and 456,000 tweets are sent. That's a colossal amount of data to process, and impossible for humans to do it alone. If machines are made solely responsible for sorting through data using text analysis models, the benefits for businesses will be huge.

For example, by using sentiment analysis companies are able to flag complaints or urgent requests, so they can be dealt with immediately – and perhaps avert a PR crisis on social media. Other uses of sentiment classifiers include assessing brand reputation, carrying out market research, and improving products with customer feedback.

Or maybe you work for Uber and you want to know what users are saying about the brand. You've read some positive and negative feedback on Twitter and Facebook. But 500 million tweets are sent each day, and Uber has thousands of mentions on social media every month. Can you imagine analyzing all of them manually? Where do you start? All this raw text data needs to be converted into numbers. This is where text analysis with machine learning plays a crucial role.

A useful option is sentiment analysis, which analyzes the opinion about a given subject within a text. By analyzing your social media mentions with a sentiment analysis model, you can automatically categorize them into Positive, Neutral or Negative. If you also analyze these mentions with a topic classifier, you can also understand what they are talking about. By running aspect-based sentiment analysis, you can automatically pinpoint the reasons behind positive or negative mentions and get precious insights.

REFERENCES

DOCUMENTATION

- Tweepy and Tweepy documentation : https://www.tweepy.org
- Twitter API : https://developer.twitter.com/en/apps/16465696
- Towards Data Science : https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-program-in-python-with-naive-bayes-classification-672e5589a7ed
- Aylien : http://blog.aylien.com/build-a-sentiment-analysis-tool-for-twitter-with-this-simple-python-script/
- Stack Overflow: https://stackoverflow.com/questions/54276075/how-to-display-the-sentiment-analysis-values-in-a-pie-chart-using-matplotlib-in
- Hackernoon : https://hackernoon.com/twitter-scraping-text-mining-and-sentiment-analysis-using-python-b95e792a4d64
- Real Python: https://realpython.com/twitter-sentiment-python-docker-elasticsearch-kibana/
- Analytics Vidhya : https://www.analyticsvidhya.com/blog/2017/03/measuring-audience-sentiments-about-movies-using-twitter-and-text-analytics/

CODE

- Jupyter Notebook
- Atom
- GitHub : https://github.com/ashjojones/sentimentanalysisTwitter

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Team Cook

Hélène LI, Kai NAKAMURA, Emeline Yisu XU