SCRIPTING VERSLAG

MoodAnalyser

Dit document is een uitleg van de Moodanalyser

Bonnie Nti Mensah
S1079357@student.hsleiden.nl
Inf4B-Software Enginering
Docent: Peter van Wijden

Inhoudsopgave

Inleiding

Dit verslag gaat over het maken van de scripting opdracht Twitter MoodAnalyser. In dit verslag ga ik stap voor stap uitleggen wat de codes doen met screenshots.

Ik maak gebruik van libraries van python en aantal zelf geschreven modules.

Moodanalyser

Opdrachtomschrijving

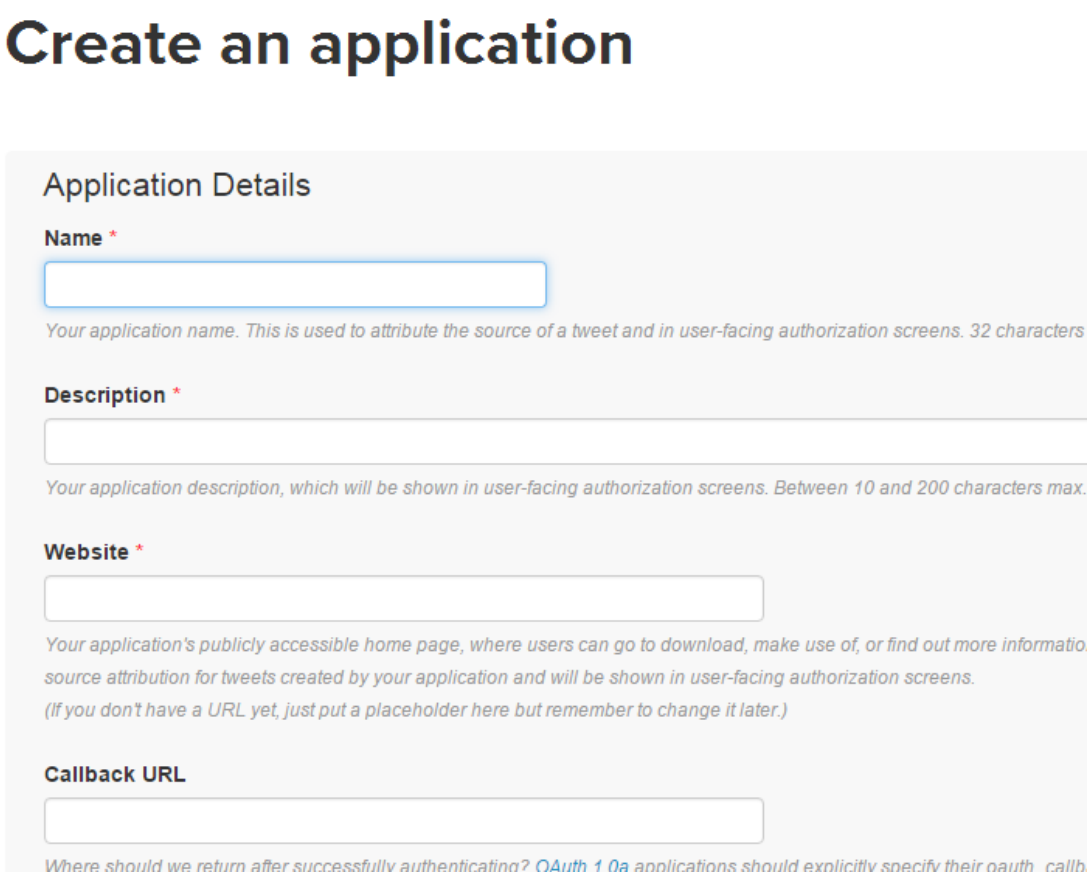
De gegeven opdracht voor scripting is een Twitter feed analyseren om te bepalen wat voor sentiment de feed heeft. Dit houdt in dat ik een Twitter mood analysis ga uitvoeren. De resultaten van de mood analysis geef ik weer in een visualisatie. Naast de basisopdracht is er ook een uitbreidingsopdrachten die specialisatie gericht is.

Stap 1 account aanmaken

De eerste stap was het registeren en een account aanmaken bij Twitter als een developer. Met deze account kon ik een app maken.

Met deze app kon ik Keys en tokens aanmaken. Hieronder zijn de Keys en token die nodig zijn om verder te gaan.

Consumer key: xxxxxxxx
Consumer secret: xxxxxxxx
Access token: xxxxxxxxxx
Access secret: xxxxxxxxxx



Stap 2 fetch Tweets

Tweede stap was het ophalen van e twitter feeds, maar om dit te kunnen doen moest ik eerst de Twitter API importen. Na een onderzoek bleek dat Tweepy een twitter API wrapper was. Hier is de implementatie ervan:

```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import time

ckey = 'L
csecret = '9ox90xOtrY                    Ihk7EHFbqLumJ5zRSw9qxUl9ypDl'
atoken = '4646906597-dxqlK5QKcUCs4r3kXPAB9VqA          06QjDvb'
asecret = '2zMcDflnk                      DbT6ur1WSUTzQJohJjfw'

class Listener(StreamListener):
    def on_data(self, data):
        tweet = data.split(',"text":"')[1].split('","source')[0]
        saveThis = str(time.time()) + '::' + tweet
        output = open('output.txt', 'a')
        output.write(saveThis)
        output.write('\n')
        output.write('\n')
        output.close()
        return True

    def on_error(self, status):
        print(status)

auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
tStream = Stream(auth, Listener())
tStream.filter(track=["Obama"])
```

Zoals te zien in dit stukje code is het geschreven in vier stukken.

Het eerste stuk maakt gebruik van de Tweepy API om met de Twitter API te communiceren. De import Stream, OAuthHandler en StreamListener.

Het tweede stuk is de invulling van de Keys en Tokens om de twitter API te authentiseren.

Vervolgens wordt een class Listener gemaakt met de parameter StreamListener. Deze class heeft twee functies on_data en on_error. De on_data functie heeft twee parameter (self, data). self is een parameter van PYTHON en data is de opgehaalde feeds van Twitter.

Een tweet van de API ziet er zo uit:

“ {"created_at":"Wed Feb 03 14:03:31 +0000 2016","id":694883919401889793,"id_str":"694883919401889793", *"text":"India's accelerating growth revs up car sales: Consumers in the world's fastest-growing major economy are kick...* https:\/\/t.co\/KsOsxRLwpb”, source":"\u003ca href=\"http:\/\/twitterfeed.com\" rel=\"nofollow\"\u003etwitterfeed\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":2152269158,"id_str":"2152269158","name":"Mutinda Kisio","screen_name":"YAHSHUAmob","location":null,"url":"http:\/\/yahshuamob.com","description":"Great Pyramid of Giza http:\/\/ow.ly\/q7y7u built 481 ft in 2560 BCE while The Tower of Babel 300ft in 610 BC http:\/\/ow.ly\/q7ycV WHO was closer to God?","protected":false,"verified":false,"followers_count":95,"friends_count":18,"listed_count":90,"favourites_count":1,"statuses_count":21580,"created_at":"Thu Oct 24 05:00:56 +0000 2013","utc_offset":null,"time_zone":null,"geo_enabled":false,"lang":"en","contributors_enabled":false,"is_translator":false,"profile_background_color":"C0DEED","profile_background_image_url":"http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile_background_image_url_https":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile_background_tile":false,"profile_link_color":"0084B4","profile_sidebar_border_color":"C0DEED","profile_sidebar_fill_color":"DDEEF6","profile_text_color":"333333","profile_use_background_image":true,"profile_image_url":"http:\/\/abs.twimg.com\/sticky\/default_profile_images\/default_profile_4_normal.png","profile_image_url_https":"https:\/\/abs.twimg.com\/sticky\/default_profile_images\/default_profile_4_normal.png","default_profile":true,"default_profile_image":true,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,"is_quote_status":false,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[],"urls":[{"url":"https:\/\/t.co\/KsOsxRLwpb","expanded_url":"http:\/\/cnn.it\/1R1JkE1","display_url":"cnn.it\/1R1JkE1","indices":[114,137]}],"user_mentions":[],"symbols":[]},"favorited":false,"retweeted":false,"possibly_sensitive":false,"filter_level":"low","lang":"en","timestamp_ms":"1454508211361"} ”

De echte tweet zelf begint bij *"text":"India's accelerating growth revs up car sales: Consumers in the world's fastest-growing major economy are kick...”*

Om te voorkomen dat het bestand waarin ik de feeds opsla niet te groot wordt filter ik de feeds door voordat ik het opsla. Een.txt bestand “output.txt” wordt aangemaakt in de map van de App.

Het laatste stukje authentiseert de Keys en tokens. tStream wordt gebruikt om de zoekterm is vinden.

Stap 3 analyse Tweets

Analyse tweet

De derde stap is het schrijven van de Analyse functie. Hiervoor heb ik een module gemaakt met een functie getTweetMood(). Deze functie heeft een paramater “tweet”.

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

stop_words = stopwords.words("english")
sentiment_neg = open("negative-words.txt","r").read()
sentiment_pos = open("positive-words.txt","r").read()
sentiment_neg = word_tokenize(sentiment_neg)
sentiment_pos = word_tokenize(sentiment_pos)


def getTweetMood(tweet):
    filtered_words = []
    words = word_tokenize(tweet)
    for word in words:
        if word not in stop_words:
            filtered_words.append(word)
    neutral = 0
    pos_words = 0
    neg_words = 0
    for word in filtered_words:
        if word in sentiment_pos:
            pos_words += 1
        elif word in sentiment_neg:
            neg_words += 1
        else:
            neutral = 0

    if pos_words == neg_words == neutral == 0:
        sentiment_value = neutral
    elif pos_words > neg_words:
        value = (pos_words * 100) /len(filtered_words)
        sentiment_value = value / 10
    else:
        value = (neg_words *-100) /len(filtered_words)
        sentiment_value = value / 10

    return sentiment_value
```

De module maakt gebruikt van een externe library de NLTK library. NLTK is Natural Language ToolKit van Twitter en met dit kon ik makkelijker zinnen manipuleren.

een list met de Engelse stopwoorden gemaakt dankzij de NLTK library, ook de bestanden “negative_words.txt” en “positive-words.txt” worden ingelezen in twee List. Deze drie lijsten met worden zijn nodig om de analyse te maken.

De getTweetMood() functie heeft een parameter “tweet”. . in de functie wordt de parameter “tweet” in een list gezet door middel van de NLTK word_tokenizer.

Elk woord in “tweet” wordt vergeleken in positieve woorden lijst en in de negatieve woorden lijst. Een counter houdt bij hoeveel negatieve of positieve woorden in een tweet bestaat om te bepalen of de tweet negatief, positief of een neutraal sentimenten heeft.

Stap 4 Read Tweets

Vierde stap is het inlezen van de Tweets nu dat ze opgeslagen zijn in een.txt bestand. Om dit te kunnen doen heb ik een Class gemaakt: “Read Tweets” met de parameter “moodAnalyser”. De Class Read Tweets heeft drie attributen en twee functies met een constructor. De class ziet er zo uit:

```
import random

class readTweets():
    __moodAnalyser = None
    __xAs = range(-10 , 11)
    __yAs = []

    def __init__(self, moodAnalyser):
        self.__moodAnalyser = moodAnalyser
        print(self.__moodAnalyser)

        twitterFeeds = open("out.txt","r").read()
        tweets = twitterFeeds.split("\n")

        for tweet in tweets:
            sentiment_value_of_tweet =  self.__moodAnalyser.getTweetMood(tweet)
            self.__yAs.append(sentiment_value_of_tweet)

    def getX():
        return self.__xAs

    def getY():
        return self.__yAs
```

In de constructor wordt het bestand gelezen en vervolgens per tweet gestuurd naar de “moodAnalyser” voor een sentiment analyse. Dit levert een waarde op en het waarde wordt opgeslagen in een array die straks gebruikt wordt in de grafiek.

De functies getY() en getX() zijn nodig om de attributen te roepen buiten de class.

Read tweets

Stap 5 teken Graph

Stap 5 is het tekenen van de grafiek. Een class graphtweet wordt gemaakt met twee parameters. Deze parameters(x, y) zijn list met coördinaten en worden gebruikt om de grafiek te tekenen. De class maakt gebruik van de Plotly library. Plotly is een library om grafieken te maken.

Deze class tekent twee grafieken een Scatter grafiek en een doorgetrokken grafiek. Bij het uitvoeren van dit stukje code worden de grafieken getekend in jouw lokale browser.

```
import plotly
from plotly.graph_objs import Scatter, Layout
import plotly.graph_objs as go


class graphTweets():

    def __init__(self, x, y):
        plotly.offline.plot({
        "data": [
            go.Scatter(
                x = x,
                y = y,
                mode = "makers"
                )
        ],
        "layout": Layout(
            title="Graphical display of Goerge Bush"
        )
        })

        # Draw scattered graph
        trace = go.Scatter(
            x = x,
            y = y,
            mode = 'markers'
        )

        data = [trace]
        plotly.offline.plot(data, filename = 'basic-scatter.html')
```