UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Clasificarea, clusterizarea si cartografierea datelor de pe Twitter folosind Apache Spark si Folium

propusă de

Rareș Bradea

Sesiunea: *Iulie, 2018*

Coordonator științific

Lect.dr. Cristian Frăsinaru

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

Clasificarea, clusterizarea si cartografierea datelor de pe Twitter folosind Apache Spark si Folium

Rareș Bradea

Sesiunea: *Iulie, 2018*

Coordonator științific
Lect.dr. Cristian Frăsinaru

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Clasificarea, clusterizarea și cartografierea datelor de pe Twitter folosind Apache Spark si Folium*” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imaginile etc. preluate din proiecte *open-source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, *data*

Absolvent *Rareș Bradea*

____

(semnătura în original)

Capitolul 1
Colectarea datelor in timp real

Orice aplicație ce include în implementarea ei și un modul de învățare automată are imperioasă nevoie sau cel puțin beneficiază foarte mult de pe urma unui dataset cât mai extins, dar și curat. Twitter, prin popularitatea sa imensă atinge cu excelență punctul referitor la cantitatea datelor, dar se îndepărtează de un ideal al datelor relevante și curate. Dat fiind faptul că oricine poate să își facă un cont unde poate să exprime idei în limita a 280 de caractere, nu este o surpriză faptul că relevanța multor tweeturi este minimă pentru multe persoane. Partea de clasificare a aplicației se va ocupa de etichetarea acelor tweeturi ce conțin enunțuri fără sens, cuvinte deosebit de vulgare, încercări de comercializare sau spam. Înainte de a ajunge acolo, trebuie să clădim un dataset cât mai mare pentru a micșora impactul postărilor cu relevanță scăzută.

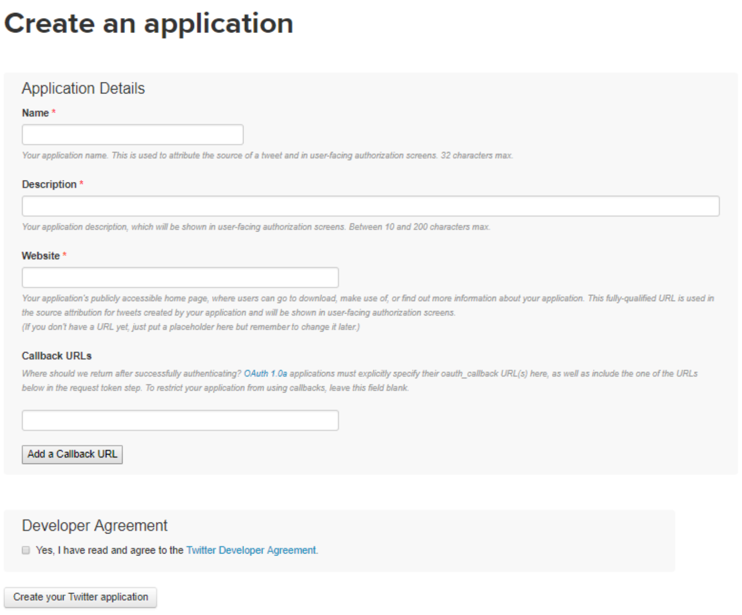
Acestea fiind spuse, Twitter este un ecosistem foarte complex si activ, iar pentru a facilita munca dezvoltatorilor de software interesați de datele ce rezidă în interiorul aplicației lor, s-a creat un API ce expune accesul la tweeturi în timp real. Aplicația mea urmărește să clasifice, clusterizeze și apoi să afișeze pe hartă tweeturi care au o vechime scurtă, astfel că este foarte utilă utilizarea API-ului de streaming oferit. Alături de acesta, voi folosi și o instanță de Apache Spark cu un context de Streaming ce permite procesarea datelor stream-uite live.

O altă variantă ar fi fost folosirea unor dataseturi deja existente, însă relevanța lor referitoare la vârstă și noutate ar fi fost discutabilă. De altfel, multe dataseturi urmăresc un singur subiect, cum ar fi tweeturi ce discută situația imigrării in Canada. Acest lucru se dovedește a fi util pentru partea de testare a clusterizării, dar scopul aplicației este să fie semi-realtime și să dispună de o varietate a subiectelor vastă și necontrolată a priori.

În cele ce urmează, voi descrie succint pașii parcurși de mine pentru a ajunge la un modul ce accesează API-ul de tweet streaming si trimite date către o instanță stream-ready de Spark ce le procesează.

În primul rând, pentru a avea acces la API-ul Twitter este nevoie un cont simplu de Twitter și mai apoi de înregistrarea unei aplicații ce dorește accesul la API, accesând pagina https://apps.twitter.com/.

Dupa completarea acestui formular, vom avea acces la un dashboard cu anumite informatii. De acolo vom prelua patru coduri importante si necesare autorizarii noastre la serviciul oferit. Acestea sunt un access token, un access token secret, consumer key si consumer key secret. Le voi folosi pentru a face autorizarea printr-un request Oauth1 folosind biblioteca 'requests_oauthlib' din Python, astfel.



Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URLs

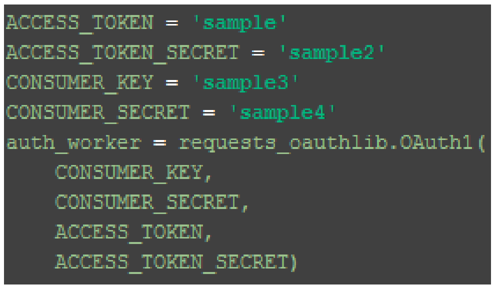
Where should we return after successfully authenticating? OAuth 1.0a applications must explicitly specify their oauth_callback URL(s) here, as well as include the one of the URLs below in the request token step. To restrict your application from using callbacks, leave this field blank.

Add a Callback URL

Developer Agreement

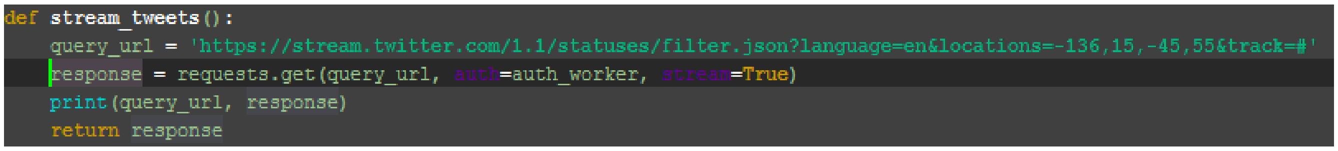
☐ Yes, I have read and agree to the Twitter Developer Agreement.

Create your Twitter application



```
ACCESS_TOKEN = 'sample'
ACCESS_TOKEN_SECRET = 'sample2'
CONSUMER_KEY = 'sample3'
CONSUMER_SECRET = 'sample4'
auth_worker = requests_oauthlib.OAuth1(
    CONSUMER_KEY,
    CONSUMER_SECRET,
    ACCESS_TOKEN,
    ACCESS_TOKEN_SECRET)
```

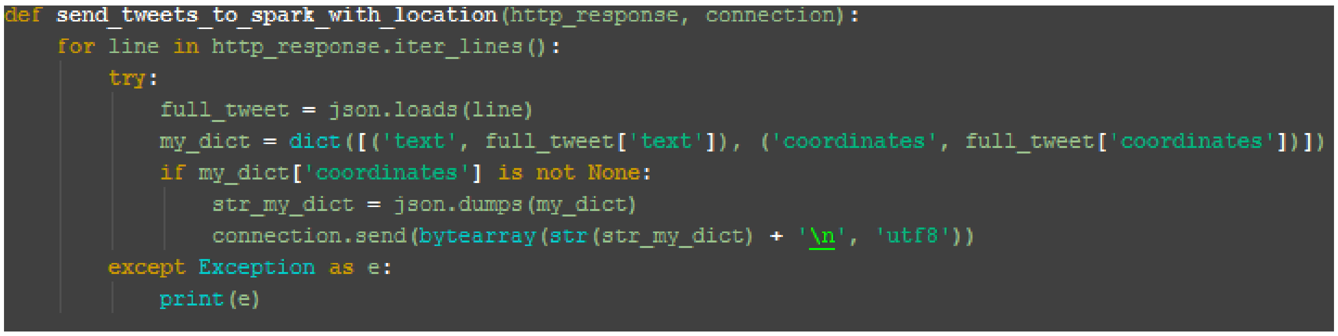
Mai departe, pornind un server care asteapta cereri pe localhost la un port oarecare (aici 9999), voi face accesul unui stream de date de la Twitter catre o instanta de Apache Spark.



```python
def stream_tweets():
    query_url = 'https://stream.twitter.com/1.1/statuses/filter.json?language=en&locations=-136,15,-45,55&track=#'
    response = requests.get(query_url, auth=auth_worker, stream=True)
    print(query_url, response)
    return response
```

Functia 'stream_tweets' returneaza un raspuns http ce contine, pentru fiecare linie, un tweet complet respectand parametrii dati in query. Mai exact, cautam tweeturi scrise in limba engleza, iar parametrul locations astfel setat ne garanteaza ca locatia mesajelor provin din aproximativ partea continentala a Statelor Unite ale Americii.

Am ales aceasta locatie deoarece Twitter este foarte popular si utilizat in respectiva tara, fapt ce denota o posibila diversitate mai mare decat in alte locuri. Subiectele discutate sunt extrem de variate si ne pot oferi multe sanse de a obtine date interesante si din prisma geolocatiei, un atribut ce il are orice tweet al carui utilizator permite accesul serviciilor de locatie asupra contului.

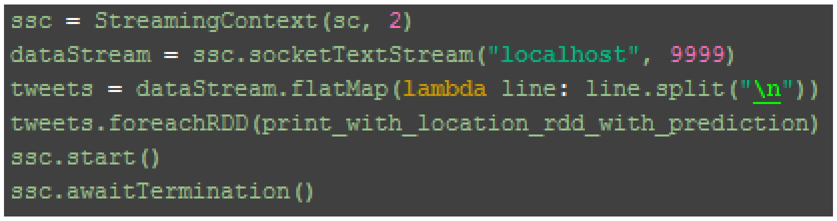


```python
def send_tweets_to_spark_with_location(http_response, connection):
    for line in http_response.iter_lines():
        try:
            full_tweet = json.loads(line)
            my_dict = dict([('text', full_tweet['text']), ('coordinates', full_tweet['coordinates'])])
            if my_dict['coordinates'] is not None:
                str_my_dict = json.dumps(my_dict)
                connection.send(bytearray(str(str_my_dict) + '\n', 'utf8'))
        except Exception as e:
            print(e)
```

Functia 'send_tweets_to_spark_with_location' primeste un raspuns http (in cazul acesta, cel returnat de functia 'stream_tweets'), o conexiune si trimite prin aceasta un vector de bytes ce reprezinta o parte dintr-un tweet. Domeniul aplicatiei ne permite sa folosim doar o parte din nenumaratele atribute ale unui tweet. Mai exact, avem nevoi doar de text si de coordonatele geografice reprezentant punctul de unde a fost trimis acel tweet.

Aceste functii fac parte dintr-un script Python care la executie face bind si listen unui socket pe localhost, port 9999. Cand primeste o cerere, accepta si trimite prin conexiune, folosind functiile descrise mai sus, tweeturi catre entitatea care face cerere.

A doua parte a procesului se refera la aceasta entitate. Ea este un DataStream creat de un StreamingContext din biblioteca pyspark.streaming. DataStreamul este descris de un socketTextStream ce face cereri la serverul numit mai sus. Pe stream aplicam o functie pe datele live pentru a face split intre tweeturi. Apoi pentru fiecare tweet afisam si salvam intr-un fisier, pentru fiecare tweet, textul, locatia precum si predictia facuta de clasificatorul de spam.



```python
ssc = StreamingContext(sc, 2)
dataStream = ssc.socketTextStream("localhost", 9999)
tweets = dataStream.flatMap(lambda line: line.split("\n"))
tweets.foreachRDD(print_with_location_rdd_with_prediction)
ssc.start()
ssc.awaitTermination()
```

Am folosit partea de Streaming din Apache spark pentru a avea acces la utilizarea in timp real a unor functii peste niste date. Aici folosesc functia 'foreachRDD' care, dupa cum reiese si din denumire, apeleaza o anumita functie pe fiecare RDD din datastream.

Functia 'print_with_location_rdd_with_prediction' afiseaza si salveaza tweeturile astfel: folosind pe intregul tweet functia 'loads' din biblioteca json, putem incarca un dictionar dintr-un string. Pentru a putea clasifica textul unui tweet ca fiind ori spam, ori non-spam, avem nevoie sa impartim stringul in mai multe substring-uri.

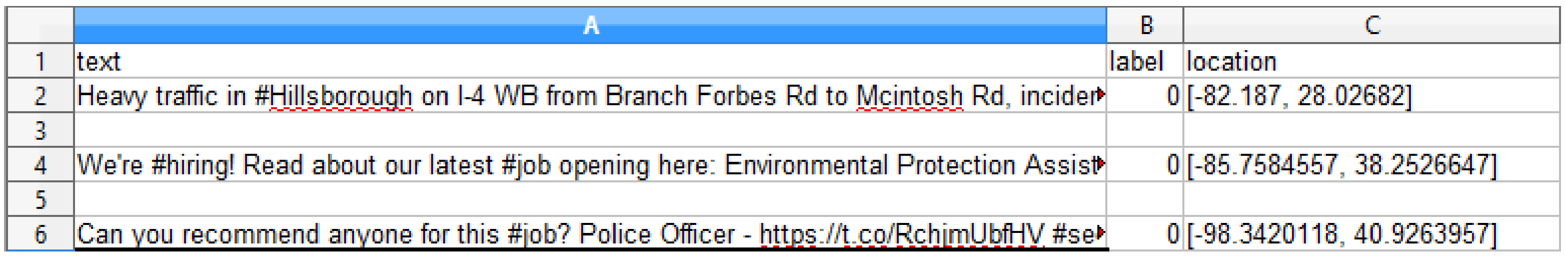
Acest proces se numeste tokenizare, iar pentru acest task se poate folosi foarte utila biblioteca creata pentru exact acest scop. Obiectul de tip TweetTokenizer din biblioteca nltk.tokenize imparte un string in tokenuri avand in vedere si structura unui tweet. Parametrii 'strip_handles' si 'reduce_len' folositi in constructorul acestei clase indeamna tokenizerul sa reduca dimensiunea tweetului daca este posibil. Acest lucru are loc daca se repeta foarte multe litere intr-un cuvant. Acest lucru este destul de des folosit, deoarece utilizatorii ar putea accentua anumite cuvinte prin repetarea unor litere.

De asemenea, referintele la alti utilizatori nu sunt incluse. Spre exemplu, textul '@remy: This is waaaaayyyy too much for you!!!!!!' este tokenizat si returnat ca intr-o lista astfel: [':', 'This', 'is', 'waaayyy', 'too', 'much', 'for', 'you', '!', '!', '!'].

Mai departe, predictia nu poate fi facuta pe niste simple cuvinte, astfel ca apelam la un procedeu numit feature hashing. Pentru fiecare term, insemnand cuvant, se calculeaza hashul acestuia si numarul de aparitii a cuvantului in text si aceste rezultate sunt pastrate intr-un SparseVector. Modelul de clasificare are nevoie de acest vector rar pentru a face predictia. Acest procedeu este detaliat in capitolul referitor la Clasificare.

Cat timp serverul detaliat in prima parte ramane pornit, acest al doilea script primeste tweeturi, le clasifica si le scrie intr-un fisier pentru a fi apoi folosit de celelalte scripturi.

In final, vom avea date care arata in felul urmator si pe care se poate lucra foarte usor spre a fi clusterizate dupa subiect si cartografiate estetic, scotand in evidenta tweeturile care au in componenta un anumit termen dat ca input.



	A	B	C
1	text	label	location
2	Heavy traffic in #Hillsborough on I-4 WB from Branch Forbes Rd to Mcintosh Rd, incider	0	[-82.187, 28.02682]
3			
4	We're #hiring! Read about our latest #job opening here: Environmental Protection Assist	0	[-85.7584557, 38.2526647]
5			
6	Can you recommend anyone for this #job? Police Officer - https://t.co/RchjmUbfHV #se	0	[-98.3420118, 40.9263957]