Real-Time Data Enrichment

1 Problem Statement

To enrich tweets content on real-time at scale through internal corpus

2 Explanation

In this case, we aim to enrich on real time the data obtained from tweets and process them dynamically to get certain necessary details from it. We try to extract the state from which user belongs who has tweeted. Furthermore, we also try to extract the hobbies of the user.

3 Literature Review

While social media have become mainstream, as shown by the ever growing number of users, Twitter stands out as the quintessential platform to openly access real-time updates on breaking news and ongoing events. With over 500 million users, Twitter sees a daily stream of more than 400 million short messages known as tweets. These tweets include from one-to-one conversations and chatter, to updates of wider interest about current affairs, encompassing all kinds of information.
Twitter has become a huge social media service where millions of users contribute on a daily basis. Two features have been fundamental in its success: (1) the shortness of tweets, which cannot exceed 140 characters, facilitates creation and sharing of messages in a few seconds, and (2) the easiness of spreading those messages to a large number of users in very little time. Throughout the time, the community of users on Twitter has established a syntax for interaction with one another, which has become the standard syntax later officially adopted by its developers. Most major Twitter clients have implemented this standard syntax as well.

4 Data

Data is obtained real time from twitter API in form of a JSON object containing metadata and data about the tweet.

5 Deliverable

1. Code including all processes from ingestion to output
2. Output i.e. State and hobbies of user

6 Evaluation

Not Applicable

7 Data Ingestion

Data is ingested in form of JSON object through spark streaming. A port 5556 is used to listen and obtain the data from the API.

```
"""
 DATA INGESTION
----------------
"""

class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            print(data.split('\n'))
            self.client_socket.send(data)
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True

    def on_error(self, status):
        print(status)
        return True
```

8 Data Analysis

Not Required

9 Data Munging

Data Munging is performed in which names are extracted, locations and date time are extracted.

```
    def __preprocessing(self, lang):
        sentences = sent_tokenize(self.sentence)
        sentences = [word_tokenize(sent) for sent in sentences]
        sentences = [pos_tag(sent, lang=lang) for sent in sentences]
        return sentences

    def extract_names(self, lang):...

    def extract_location(self, lang):...

    def extract_date_time(self):
        dateTimes = []
        grammar = r"DATE: {<NNP><CD>}"
        parser = nltk.RegexpParser(grammar)

        phrase_tagger = nltk.pos_tag(word_tokenize(self.sentence))
        phrase_chunk = nltk.ne_chunk(phrase_tagger)

        phrase_date = parser.parse(phrase_chunk)
        for word in phrase_date:
            if isinstance(word, nltk.tree.Tree) and word.label() == 'DATE':
                dateTimes.append(' '.join([w[0] for w in word]))
        return dateTimes
```

Next we try to extract the state name from the tweet and finally the hobby of the person. We search for places, zip codes or state names to get the name of the state the user belongs to.
For hobby, we match the tweet from corpus of hobbies we created to check if user has any hobby from it.

```
"""
 DATA ENRICHMENT
-----------------
"""
def searchGeolocation(sentence): ...

def stateSearch(sentence): ...

def zipcodeExtracter(sentence): ...

def get_result(sentence): ...

def map_tweets(tweet):

    json_tweet = json.loads(tweet)
    if json_tweet.has_key('lang'): # When the lang key was not presen
        if json_tweet['lang'] == 'en':
            tweet = json_tweet['text']
            # user_mentionList_dict1 = json_tweet['entities']
            print "\n\n\n\n"
            print json_tweet
            print "\n\n\n\n"

            print "**************************************************

            print "\n\n\n\n"
            # print user_mentionList_dict1,'\n\njfyhdhgdhfhnuyjfutdju
            # user_mentionList_dict2 = user_mentionList_dict1['user_m
            name = json_tweet['user']['name']
            userid = json_tweet['user']['id']
            import pandas as pd
            referance = pd.read_csv('../Case/us_codes.csv')

            statelist = referance['State'].tolist()

            stateabbrv = referance['State Abbreviation'].tolist()

            countrylist = referance['County'].tolist()

            placelist = referance['Place Name'].tolist()

            ziplist = referance['Zip Code'].tolist()

            latlist = referance['Latitude'].tolist()

            longlist = referance['Longitude'].tolist()
```

10 Data Exploration

Not Required

11 Feature Engineering

Not Required

12 Modeling

Not Applicable

13 Optimization

Not Applicable

14 Prediction

Not Applicable

15 Visual Analysis

Not Applicable

16 Results

UserID	State	Hobbies
8.26815258270564E+017	-----	-------
7.00434197219902E+017	-----	u'Tenni'
8.04766204799943E+017	-----	-------
2682553140	-----	-------
8.28583568880329E+017	-----	-------
8.4065789600639E+017	-----	-------
4794913036	-----	-------
2411802678	-----	-------
20435517	-----	-------
8.04766204799943E+017	-----	u'Magic'
3436190953	-----	-------
7.65137770322985E+017	-----	-------
210187178	-----	u'Go'
9.17116964258111E+017	New York	-------
4651831592	-----	-------
7.63906257409315E+017	-----	-------
27366865	-----	-------
377663392	-----	-------
522431966	-----	-------
9.61007846690382E+017	-----	-------
2517225447	-----	-------
1458887952	-----	-------
288864844	-----	-------
7.54653348180156E+017	Georgia	-------
273537714	-----	-------
8.49807575235408E+017	-----	-------
7.63906257409315E+017	-----	-------
2348985279	-----	-------
7.63906257409315E+017	New York	-------
4210104130	-----	-------
9.14131338206122E+017	-----	-------
973238718	-----	-------
8.32003294205731E+017	-----	-------
85006931	-----	-------
555890683	-----	-------
8.10885846471614E+017	-----	-------
7.24752198236787E+017	-----	-------
33096452	-----	-------
38252077	California	-------