PS2_zho15

October 29, 2021

1 Project 1

Name: Diego Aldo Pettorossi

abc123: zho125

1.1 Introduction

You are a data scientist working for the government. You want to understand the public opinion regarding hurricane Maria which is responsible for killing at least 499 people in Puerto Rico. Total losses are estimated at $94.4 billion dollars which accrued to government agencies, businesses, and more importantly, familes [1]. With this background, whether you are a politician, bussiness person, or one effected by the hurricane, understanding the sentiment of the general populace is important. For this assigment, you will use a subset of the tweets retrieved from Twitter that mentioned #PuertoRico over the period of October 4 to November 7, 2017 [2] to measure the sentiment (i.e., the "good" or "bad" opinions people have about the hurricane and its impact) of this event.

For this task, we will write code for a lexicon-based analysis (i.e., lexicon-based classification). Lexicon-based classification is a way to categorize text based using manually generated lists of topical words. Essentially, we just need to check if the topical words appear in a piece of text (e.g., a tweet). In this projectwe will make use of manually curated sentiment words. However, the basic experimental process is the same for other tasks (e.g., identifying offensive language).

[2] site: https://archive.org/details/puertorico-tweets

1.2 You should write code that does the following:

1. Keeps track of the number of tweets
2. Keeps track of the number of positive and negative tweets
3. Keeps track of the user that tweets the most
4. Keeps track of the total number of unique user
5. Keeps track of the average number of tweets per user
6. Keeps track of the most positive and negative tweets

[1]:
```
def file_to_set(file):
    #textReader= file.readlines() #read file as a text
    setWords=set()

    for line in file:    #line is assigned to a row on each loop cycle
```

```
        setWords.add(line.strip())   #add the word to the set (if not present␣
 ↪already)

    return setWords    #set of words

positive_file = open('./bing_liu/positive-words.txt', encoding='utf8')
positive_words = file_to_set(positive_file) # the function is taking a file␣
 ↪handle as input.
positive_file.close()

negative_file = open('./bing_liu/negative-words.txt', encoding='iso-8859-1') #␣
 ↪If you get a weird read error. Let me know. We can change the encoding.
negative_words = file_to_set(negative_file)
negative_file.close()
```

[2]:
```
assert(type(positive_words) == type(set()))
assert(type(negative_words) == type(set()))
assert(len(positive_words) == 2006)
assert(len(negative_words) == 4783)
assert(('good' in positive_words)  == True)
assert(('bad' in negative_words)  == True)
assert(('bad' not in positive_words) == True)
print("Asserts finished successfully!")
```

```
Asserts finished successfully!
```

[3]:
```
def count_sentiment_words(sentiment_set, tweet_text, lower):
    WordsCount = 0
    if lower:     #check lower value
        tweet_text = tweet_text.lower() #if True lowercase the string
    listTweetText = tweet_text.split() #create a list, so I can index it later
    # print(listTweetText) #check the list
    for indexLoop in listTweetText:
        if indexLoop in sentiment_set: #check if the word is in the set
            WordsCount += 1
            # print(WordsCount) #number of words found so far
    return WordsCount  #total number of words found
```

[4]:
```
assert(count_sentiment_words(positive_words, "this is a good good good class",␣
 ↪True) == 3)
assert(count_sentiment_words(positive_words, "this is a good\tgood\tgood␣
 ↪class", True) == 3)
assert(count_sentiment_words(positive_words, "this is a GOOD GOOD GOOD class",␣
 ↪False) == 0)
assert(count_sentiment_words(positive_words, "this is a GOOD GOOD GOOD class",␣
 ↪True) == 3)
```

```
assert(count_sentiment_words(positive_words, "this is a GOOD GOOD good class",␣
 ↪False) == 1)
assert(count_sentiment_words(positive_words, "Python is the best programming␣
 ↪language for data science", True) == 1)
assert(count_sentiment_words(negative_words, "R is bad compared to Python ;)",␣
 ↪True) == 1)
print("Asserts finished successfully!")
```

```
Asserts finished successfully!
```

[5]:
```
def predict(num_pos_words, num_neg_words):

    prediction = ""
    if num_pos_words > num_neg_words: #compare the number of positive and␣
 ↪negative words
        prediction = "positive"
    elif num_pos_words < num_neg_words:
        prediction = "negative"
    else:
        prediction = "neutral"

    return prediction
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[6]:
```
assert(predict(2, 5) == 'negative')
assert(predict(5, 2) == 'positive')
assert(predict(3, 3) == 'neutral')
print("Assert finished successfully!")
```

```
Assert finished successfully!
```

[7]:
```
def predict_score(num_pos_words, num_neg_words):

    sentiment_score = None
    sentiment_score = num_pos_words - num_neg_words # difference between number␣
 ↪of positive and negative words

    return sentiment_score
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[8]:
```
assert(predict_score(3, 1) == 2)
assert(predict_score(2, 2) == 0)
assert(predict_score(2, 5) == -3)
```

```
print("Asserts finished successfully!")
```

```
Asserts finished successfully!
```

[9]:
```
import json

def json_string_to_dictionary(json_string):

    myJsonDict = json.loads(json_string) #json string -> dictonary

    return myJsonDict
```

[10]:
```
data = json_string_to_dictionary('{"a": 1}')
assert(data == {'a': 1})
data = json_string_to_dictionary('[1,2,3]')
assert(data == [1,2,3])
print("Assert finished successfully!")
```

```
Assert finished successfully!
```

[11]:
```
total_number_of_tweets = 0
total_number_of_positive_tweets = 0
total_number_of_negative_tweets = 0
lowest_sentiment_score = 0
highest_sentiment_score = 0
most_positive_tweet = "None"
most_negative_tweet = "None"
maxTweetperUser = 0
user_with_most_tweets = ""
average_number_tweets_per_user = 0
total_number_of_users = 0
UserDict = {}

twitter_dataset = open('puerto-rico.jsonl', 'r')

for row in twitter_dataset:
    tweet_dict = json_string_to_dictionary(row)

    ###############################

    tweet_text = tweet_dict['full_text']  # get "full_text" from the tweet_dict

    screen_name = tweet_dict['user']['screen_name']   # get "screen_name" from␣
 ↪the tweet_dict

    ###############################

    num_pos_words = count_sentiment_words(positive_words, tweet_text, True)
    num_neg_words = count_sentiment_words(negative_words, tweet_text, True)

    sentiment_prediction = predict(num_pos_words, num_neg_words)
    sentiment_score = predict_score(num_pos_words, num_neg_words)

    #######################

    total_number_of_tweets += 1    # update numbers of tweets

    if sentiment_score > 0:         # check if the tweet is positive
        total_number_of_positive_tweets += 1 # update the count
        if sentiment_score > highest_sentiment_score:
            highest_sentiment_score = sentiment_score
            most_positive_tweet = tweet_text

    if sentiment_score < 0:        #check if the tweet is negative
        total_number_of_negative_tweets += 1 # update the count
        if sentiment_score < lowest_sentiment_score:
            lowest_sentiment_score = sentiment_score
            most_negative_tweet = tweet_text

    if screen_name in UserDict:     # check if the user is new
        UserDict[screen_name] += 1
        if UserDict[screen_name] > maxTweetperUser:     # is the user with most␣
 ↪tweets?
            maxTweetperUser = UserDict[screen_name]     # set a new max
            user_with_most_tweets = screen_name  # set the new user with most␣
 ↪tweets

    else:
        total_number_of_users += 1   # update the count of new users
        UserDict[screen_name] = 0    # new user pair

    ################################


average_number_tweets_per_user =  total_number_of_tweets /␣
 ↪total_number_of_users # calculate average

twitter_dataset.close()
```

[12]:
```
print("Total Number of Tweets: {}".format(total_number_of_tweets))
print("Total Number of Positive Tweets: {}".
 ↪format(total_number_of_positive_tweets))
print("Total Number of Negative Tweets: {}\n".
 ↪format(total_number_of_negative_tweets))
```

```python
print("Most Positive Tweet")
print(most_positive_tweet)
print()

print("Most Negative Tweet")
print(most_negative_tweet)
print()

print("Total Number of Users: {}".format(total_number_of_users))
print("Average Number of Tweets per User: {}".
 ↪format(average_number_tweets_per_user))
print("User with the most tweets: {}".format(user_with_most_tweets))
```

```
Total Number of Tweets: 737153
Total Number of Positive Tweets: 178205
Total Number of Negative Tweets: 163502

Most Positive Tweet
@thechew @JoAnnaLGarcia @KevinProbably Such amazing #'s of precious people fed
in #PuertoRico by delicious, heartfelt work of @chefjoseandres &amp; friends.
Also, top efforts by talented chefs towards precious people in #winecountryfires
area. #northbayfires  Wish you'd have reported on it more. #SF #SonomaStrong

Most Negative Tweet
@realDonaldTrump LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR
LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR LIAR #PuertoRico #PresidentLoco

Total Number of Users: 286975
Average Number of Tweets per User: 0
User with the most tweets: Noti_PuertoRico
```

The lines below give example inputs and correct outputs using asserts, and can be run to test the code. Passing these tests is necessary, but **NOT** sufficient to guarantee your implementation is correct. You may add additional test cases, but do not remove any tests.

[35]:
```python
assert(isinstance(total_number_of_tweets, int) or␣
 ↪isinstance(total_number_of_tweets, float))
assert(isinstance(total_number_of_positive_tweets, int) or␣
 ↪isinstance(total_number_of_positive_tweets, float))
assert(isinstance(total_number_of_negative_tweets, int) or␣
 ↪isinstance(total_number_of_negative_tweets, float))
assert(isinstance(most_positive_tweet, str))
assert(isinstance(most_negative_tweet, str))
assert(isinstance(user_with_most_tweets, str))
assert(total_number_of_tweets == 737153)
print("Assert finished successfully!")
```

```
Assert finished successfully!
```