PS2

March 25, 2022

1 Problem Set 2

Name: Ashley Cortez

abc123: gql524

Collaborators: - Colaborator 1 : Rosa Martinez - Colaborator 2

1.0.1 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 exercise we will make use of manually curated sentiment words. However, the basic experimental process is the same for other tasks (e.g., identifying offensive language).

If you are interested, though it is not needed, you can learn more about lexicon-based classification in Chapter 21 (21.2 and 21.6) of the free online book at the following link: Speech and Language Processing

1.0.2 References

[1] Spalding, Rebecca (November 13, 2017). "Puerto Rico Seeks $94 Billion in Federal Aid for Hurricane Recovery". Bloomberg News. Retrieved December 15, 2017.

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

1.1 Submission Instructions

After completing the exercises below, generate a pdf of the code **with** outputs. After that create a zip file containing both the completed exercise and the generated PDF/HTML. You are **required** to check the PDF/HTML to make sure all the code **and** outputs are clearly visible and easy to read. If your code goes off the page, you should reduce the line size. I generally recommend not going over 80 characters.

For this task, unzip and move the file "puerto-rico.jsonl" in to the same directory as this notebook, then complete the following exercises. However, when you turn the assigment in, do **NOT** include puerto-rico.jsonl in your zip file when you submit the homework, you will kill Blackboard.

Finally, name the zip file using a combination of your the assigment and your name, e.g., ps2_rios.zip

1.2 Exercise 1 (1 points)

The files "positive_words.txt" and "negative_words.txt" contain mannually curated positive (e.g., good, great, awesome) and negative words (e.g., bad, hate, terrible). The files contain one word on each line. Write a function that takes the open file (i.e., the file handle) and adds the words (i.e., on each line) to a set then returns the set.

Note: You should use ".strip()" to remove the newline character from the end of each word.

[ ]:
```
# Write code here

def file_to_set(file):
    """
    This function should take a file handler as input and return a set.

        Parameters:
            - file file handle: This variable is a file handle

        Return:
            - The file should return a set (e.g., {'good', 'great', 'happy'})
    """
    # Write code here
    dataset = set()
    for i in file:
        dataset.add(i.strip())

    return dataset # You should return a set

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()
```

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.

```
[ ]: 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!
```

1.3 Exercise 2 (1 points)

For this exercise, you need to write a function that counts the number of words in a sentence that also appear in a set. For example, given the set set([‘good’, ‘great’]) and the sentence “this is good good good”, the function should return 3. The lower parameter should lowercase the input tweet text if it is set to True. Otherwise, it should keep the tweet text as-is.

Hint: You can check if something is in a set using the following notation:

```
mySet = set(["a", "b", "c"])
otherList = ["c", "d"]
for letter in otherList:
    if letter in mySet:
        print(letter)
```

The above code will print “c”.

```
[ ]: def count_sentiment_words(sentiment_set, tweet_text, lower):
         """
         This function takes a set and string as input, then counts the number of␣
     ↪words that
         appear in the string (tweet_text) that are also in the set (sentiment_set).␣
     ↪The
         tweet_text should be normalized based on the lower argument (i.e.,␣
     ↪lowercase if True)

             Parameters:
                 - sentiment_set set: A set of sentiment words, e.g., {'good',␣
     ↪'great', 'happy'}
                 - tweet_text string: A tweet, e.g., "I go to UTSA!!!"
                 - lower bool: A True or False boolean value indicating the␣
     ↪tweet_text should be lowercased. If False, keep tweet_text as-is.
         """
         # Your code here
         count = 0
```

```
    if lower: tweet_text = tweet_text.lower()
    words =  tweet_text.split()

    for word in words:
        for sentiment in sentiment_set:
            if sentiment in word: count += 1


    return count #You should return a number
```

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.

[ ]:

```
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)
```

1.4 Exercise 3 (1 point)

For this exercise, you will write a function that takes two numbers as input and returns a string. Intuitively, this is a basic classification function for lexicon-based sentiment classification.

The function should take as input parameters the the number of positive (num_pos_words) and negative (num_neg_words) words in each tweet to predict sentiment. If the number of positive words is greater than to the number of negative tweets (num_pos_words > num_neg_words), then predict **"positive"**. If the number of negative words is greater than the number of positive words (num_neg_words > num_pos_words), then predict **"negative"**. If both num_pos_words and num_neg_words are equal (num_neg_words = num_pos_words), predict **"neutral"**. This is known as lexicon-based classification.

Intuitively, the idea is simple, a tweet with more positive words is generally express "positive" sentiment. Likewise, a tweet with more negative words is expressing "negative" sentiment.

[ ]:

```
def predict(num_pos_words, num_neg_words):
    """
    This function should return the string "positive", "negative", or "neutral"␣
↪given
    the input parameters, i.e., if num_pos_words is greater than num_neg_words,␣
↪return "positive"

        Parameters:
            - num_pos_words int: This is a count representing the number of␣
↪positive words in a tweet.
            - num_neg_words int: This is a count representing the number of␣
↪negative words in a tweet.

        Return:
            - Return a string "positive", "negative", or "neutral"
    """
    # Your code here
    if num_pos_words > num_neg_words:
        return "positive"
    elif num_pos_words < num_neg_words:
        return "negative"
    else:
        return "neutral" # You should return a string
```

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.

[ ]:

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

```
Assert finished successfully!
```

1.5 Exercise 4 (1 point)

This exercise is similar to Exercise 3. However, instead of making a prediction, we should write a function that returns a sentiment score. Specifically, assume num_pos_words is 3 and num_neg_words is 4, the function should return -1. The idea is that the more *positive* the number, the more positive the sentiment. Likewise, the more *negative* the number, the more negative the sentiment.

[ ]:

```
def predict_score(num_pos_words, num_neg_words):
    """
    This function should generate a sentiment score num_pos_words -␣
↪num_neg_words.

        Parameters:
            - num_pos_words int: This is a count representing the number of␣
↪positive words in a tweet.
            - num_neg_words int: This is a count representing the number of␣
↪negative words in a tweet.

        Return:
            - Return an integer representing the difference between positive␣
↪words and negative words.
    """
    # Your code here
    return num_pos_words - num_neg_words # You should return a number
```

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.

[ ]:

```
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!
```

1.6 Exercise 5 (1 point)

Write a function that takes a json string as input and returns a Python object. Hint: This can be one line. You can use the json library.

[ ]:

```
import json

def json_string_to_dictionary(json_string):
    """
    This function should take a json string and convert it into a Python object␣
↪(hint: loads)

        Parameters:
            - json_string str: A json string, e.g., '{"a": 1}'

        Returns:
            - Return a python object, e.g., {"a": 1} <--- look no quotes, so␣
↪this is a dictionary
    """
    # Your code here
    return json.loads(json_string)
```

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.

[ ]:

```
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!
```

1.7 Exercise 6 (3 points)

For this task, we combine the functions written for the previous exercises to classify all of the tweets in a real Twitter dataset. 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 users 5. Keeps track of the average number of tweets per user (how many tweets does each user make, on average) 6. Keeps track of the most positive and negative tweets.

Note: This task depends on Exercises 1 through 5. You will need to complete them first. Also, do **not** store all of the tweets in a list. This will use too much memory because of the size of the dataset. It is okay to store all of the user's screen names.

Finally, the dataset is big! So, I recommend working on a subset of the dataset to make sure your code works, i.e., you could "break" after the first 100 lines.

[ ]:

```
total_number_of_tweets = 0
total_number_of_positive_tweets = 0
total_number_of_negative_tweets = 0
most_positive_tweet = ''
most_negative_tweet = ''
user_with_most_tweets = ''
average_number_tweets_per_user = 0
total_number_of_users = 0
users = {}
sentiment = {}
max_score = 0
min_score = 0
maxTweets = 0

# NOTE: You may need to define extra variables to help generate the required␣
 ↪output.

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


# NOTE: Process the data line-by-line. Do NOT load the entire dataset into␣
 ↪memory.

for row in twitter_dataset:
    tweet_dict = json_string_to_dictionary(row)
    if total_number_of_tweets >= 100:
        break

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

    tweet_text = tweet_dict["full_text"] # MODIFY THIS LINE TO GET THE␣
↪"full_text" from the tweet_dict
    screen_name = tweet_dict["user"]["screen_name"] # MODIFY THIS LINE TO GET␣
↪THE "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)



    #################################
    # Your code should do the following:
    #   1. Keep track of the number of tweets
    #   2. Keep track of the number of positive and negative tweets
    #   3. Keep track of the user that tweet the most
    #   4. Keep track of the total number of unique users
    #   5. Keep track of the average number of tweets per user (how many tweets␣
↪does each user make, on average)
    #   6. Keep track of the most positive and negative tweets.

    # YOUR CODE HERE

    total_number_of_tweets += 1
    if sentiment_prediction == "negative":
        total_number_of_negative_tweets += 1
    elif sentiment_prediction == "positive":
        total_number_of_positive_tweets += 1

    users[screen_name] = users.get(screen_name, 0) + 1
    sentiment[tweet_text] = sentiment.get(tweet_text, float(sentiment_score))

for k,v in users.items():
    if v > maxTweets:
        maxTweets = v
        user_with_most_tweets = k
```

```python
for i,j in sentiment.items():
    if j < min_score:
        min_score = j
        most_negative_tweet = i
    elif j > max_score:
        max_score = j
        most_positive_tweet = i


total_number_of_users = len(users)
average_number_tweets_per_user = total_number_of_tweets / total_number_of_users

twitter_dataset.close()
```

[ ]:

```python
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))

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: 100
Total Number of Positive Tweets: 30
Total Number of Negative Tweets: 59

Most Positive Tweet
RT @WesSmith123: Welcome To Day 257 Of Trump's ShitShow

#PuertoRico: No food-No water-No power
#Trump: You're lucky you didn't have a "r…

Most Negative Tweet
RT @CarlosLoret: #Trump "dando ayuda" en #PuertoRico.  El Presidente de EEUU
aventaba papel. Lamentable. #Contraportada  https://t.co/f2Q0C…
```

```
Total Number of Users: 93
Average Number of Tweets per User: 1.075268817204301
User with the most tweets: musicandpost
```

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.

[ ]:

```
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!")
```

1.8 Exercise 7 (2 points)

For this exercise, you will perform manual analysis of the predictions. Modify the code to load the tweet text, then answer the questions below.

[ ]:

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

num_tweets_to_print = 20

num_tweets = 0

for row in twitter_dataset:
    num_tweets += 1
    tweet_dict = json_string_to_dictionary(row)

    ################################
    # YOUR CODE HERE
   # MODIFY THIS LINE TO GET THE "full_text" from the tweet_dict

    tweet_text = tweet_dict['full_text'] # MODIFY THIS LINE TO GET THE␣
 ↪"full_text" 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)

    print("Tweet {}: {}".format(num_tweets, tweet_text))
    print("Tweet {} Prediction: {}".format(num_tweets, sentiment_prediction))
    print()

    if num_tweets == num_tweets_to_print:
        break

twitter_dataset.close()
```

```
Tweet 1: RT @TheSWPrincess: @bri_sacks To find out how to help, visit the site
below. Virgin Islanders are not getting the media attention that #Pue…
Tweet 1 Prediction: negative

Tweet 2: I have yet to be able to express my thoughts without expletives about
@realDonaldTrump + Hurricane Maria recovery #PuertoRico
Tweet 2 Prediction: positive

Tweet 3: RT @TalbertSwan: @TaIbertSwan @realDonaldTrump "Sire, the people don't
have power, food, or water!"

#Trump: "Let them eat paper towels!"…
Tweet 3 Prediction: positive

Tweet 4: RT @NYPDSpecialops: #NYPD ESU K9 "Harley" &amp; "Nash" deployed as part
@fema NY-TF1 have been hard at work assisting in the #PuertoRico rescue…
Tweet 4 Prediction: negative

Tweet 5: RT @StarrMSS: .@elvisduran gave 30K to @Bethenny to charter  plane to
bring supplies to #PuertoRico #HurricaneMaria. He also gave 100K to @…
Tweet 5 Prediction: negative

Tweet 6: RT @ericbolling: When will @realDonaldTrump catch a break from fake
news outrage? Very unfair slams over #PuertoRico visit.
Tweet 6 Prediction: negative

Tweet 7: FCC approves up to $77 million to restore communications after
hurricane https://t.co/hn0WqJiE9T #WonkAmerica https://t.co/m6P6RvDkZi
Tweet 7 Prediction: positive

Tweet 8: "@daddy_yankee,#PuertoRico native,to #Donate $250,000 to #Habitat &amp;
raise $1.5+ #Million!"

https://t.co/32kjy93dNZ
https://t.co/15bza8gjW0
Tweet 8 Prediction: neutral
Tweet 9: RT @ericbolling: When will @realDonaldTrump catch a break from fake
news outrage? Very unfair slams over #PuertoRico visit.
Tweet 9 Prediction: negative

Tweet 10: RT @chefjoseandres: Forget politics forget pundits. What I have seen
in #PuertoRico is people coming together, sacrificing 2 serve. This is…
Tweet 10 Prediction: neutral

Tweet 11: RT @mercycorps: Our neighbors in #PuertoRico are resilient, but they
need our help to recover + rebuild. We invite you to join us.

https:/…
Tweet 11 Prediction: positive

Tweet 12: RT @StopTrump2020: At least 34 dead - #Trump blames #PuertoRico for
#FEMA not having enough money.  #SAD! https://t.co/DHuW7xGlOY
Tweet 12 Prediction: negative

Tweet 13: RT @SamaritansPurse: With your support, our disaster response team
continues to bring emergency relief to families in #PuertoRico. https://…
Tweet 13 Prediction: negative

Tweet 14: RT @usairforce: 4 @USARMY Pave Hawks, 4 pallets of search &amp; rescue
gear, 1 ATV and 39 search &amp; rescue passengers aboard a C5 headed to #Pue…
Tweet 14 Prediction: negative

Tweet 15: RT @RoseAnnDeMoro: RNRN and @AFLCIO send 300+ volunteers to
#PuertoRico: union nurses, construction and transportation workers fly out toda…
Tweet 15 Prediction: neutral

Tweet 16: RT @daddy_yankee: I know the reconstruction of my home island will
requiere long-term solutions. - go to the link and help me raise more mo…
Tweet 16 Prediction: negative

Tweet 17: RT @Jenniffer2012: Thanks from my heart to @FLOTUS for your caring and
your commitment to help. #PuertoRico    https://t.co/p8fkXfKbXd
Tweet 17 Prediction: positive

Tweet 18: RT @RichardMadan: Here is President Trump tossing paper towels at
hurricane victims in #PuertoRico https://t.co/JjLMRNFcAt
Tweet 18 Prediction: negative

Tweet 19: RT @JimmyPatronis: I'm deploying law enforcement assets to join
@fdlepio, @FLHSMV and @MyFWC to help in #PuertoRico after #Maria: https://t…
Tweet 19 Prediction: negative

Tweet 20: RT @ExDemLatina: .@CarmenYulinCruz is a Lying policial Corrupt hack!
She has time to make another shirt for media rounds. #PuertoRico #San…
```

```
Tweet 20 Prediction: negative
```

Complete the following tasks:

- Manually annotate all of the tweets printed above. This is your interpretation. We want to understand how accurate the predictions are at the tweet-level:
    1. Tweet 1 Sentiment: Negative
    2. Tweet 2 Sentiment: Neutral
    3. Tweet 3 Sentiment: Neutral
    4. Tweet 4 Sentiment: Negative
    5. Tweet 5 Sentiment: Positive
    6. Tweet 6 Sentiment: Negative
    7. Tweet 7 Sentiment:Positive
    8. Tweet 8 Sentiment: Positive
    9. Tweet 9 Sentiment: Negative
    10. Tweet 10 Sentiment: Neutral
    11. Tweet 11 Sentiment: Positive
    12. Tweet 12 Sentiment: Negative
    13. Tweet 13 Sentiment: Negative
    14. Tweet 14 Sentiment: Negative
    15. Tweet 15 Sentiment: Neutral
    16. Tweet 16 Sentiment: Negative
    17. Tweet 17 Sentiment: Positive
    18. Tweet 18 Sentiment: Negative
    19. Tweet 19 Sentiment: Negative
    20. Tweet 20 Sentiment: Negative
- How many of the predictions are right or wrong compared to your annotations?
    - 14
- Do you see any major limitations of lexicon-based classificaiton (i.e., making sentiment predictions using individual words)? Use your intuition, I will accept most answers, as long as it makes some sense. Please describe and provide examples below:
- lexicon based classification isn't very accurate considering the different ways people perceive something to be positive or negative.

Answer Here

1.9 Extra Credit (2 points)

For this exercise, you should use a different dataset (email me if you want me to share a dataset with your directly, or you can use your own—see below for dataset resources) and analyze it with a different (non-sentiment) lexicon. You can complete the same analysis as above, or do something different as long as you make use of a new lexicon and a new dataset. Notice some lexicons only have one class, or more than 2 classes. The processing will be slightly different. Feel free to ask how this can be done via Slack.

Possible lexicons: - Hate speech and offensive language lexicons: - https://github.com/steve050798/hate-speech-and-offensive-language/raw/master/lexicons/hatebase_dict.csv - https://www.cs.cmu.edu/~biglou/resources/bad-words.txt - Emotion Lexicon: - https://saifmohammad.com/WebPages/NRC-Emotion-

Lexicon.htm#:~:text=The%20NRC%20Emotion%20Lexicon%20is,were%20manually%20done%20by%20crowdsou
- Note that this lexicon has more than two classes. Message me for discussion about how to handle this.

You may use one of the lexicons above or another non-sentiment lexicon of your choice. I may allow another sentiment lexicon if the new analysis is interesting; however, email me before doing this. An example would be using a Spanish sentiment lexicon to analyze Spanish text, or doing a different analysis than the one abouve (e.g., anayzing sentiment over time).

Where to look for new data? - https://www.kaggle.com - https://www.docnow.io/catalog/ (Some datasets contain the text, not just tweet ids). If you have something specific you want, I can grab it for you using my developer account. - Google :). You do not need to use tweets.

You can pull new data from Twitter, but you will need to create a developer account. The easiest way to pull new data is via the use of Twarc (https://github.com/DocNow/twarc).

Finally, when you submit, I don't need to have your data, just make sure to provide a couple of examples, a link to the data (if available), and make sure all of the output of your code is printed with the output so I can analyze it.

[ ]:
```
# Write and document your code here
```