MIE 1624 Introduction to Data Science and Analytics – Winter 2017

Assignment 1

Due Date: 6:00pm, February 20, 2017

Submit via Blackboard

Background

Sentiment Analysis is a branch of Natural Language Processing (NLP) that allows us to determine algorithmically whether a statement or document is "positive" or "negative".

Sentiment analysis is a technology of increasing importance in the modern society as it allows individuals and organizations to detect trends in public opinion by analyzing social media content. Keeping abreast of socio-political developments is especially important during periods of policy shifts such as election years, when both electoral candidates and companies can benefit from sentiment analysis by making appropriate changes to their campaigning and business strategies respectively.

The purpose of this assignment is to compute the sentiment of text information, in our case tweets posted during the 2016 Canadian elections, and answer the question regarding: ***"Can we use Sentiment analysis on Twitter data to get an insight into the Canadian political landscape?"***

Central to sentiment analysis are techniques first developed in text mining. Some of those techniques require a large collection of classified text data often divided into two types of data, a training data set and a testing data set. The training data set is further divided into data used solely for the purpose of building the model and data used for validating the model. The process of building a model is iterative, with the model being successively refined until an acceptable performance is achieved. The model is then used on the testing data in order to calculate its performance characteristics, for example accuracy, sensitivity and specificity.

The classified data set for this assignment is a file called *classified_tweets.txt* containing tweets that have had their sentiments already analyzed and recorded as binary values, 0 and 1. A value of 0 is a negative tweet and a value of 1 is a positive tweet. For this assignment, the unclassified data set to be analyzed is the *unclassified_tweets.txt*, which contains a list of unclassified tweets from the last Canadian election.

Learning Objectives

- How to parse and clean data
- How to perform exploratory analysis on a given a data set by apply Big Data techniques to analyzing text data
- How to write and implement algorithms
- How to analyze an algorithm
- How to analyze and display results

Tools Required

- **Software**
  - Python Version 2.7 is only allowed. Python 3 **cannot** be used for this assignment
  - Python can be run from your local pc or DataScienceWorkbench website
- **Libraries**
  - Numpy
  - **NOTE: No other libraries are allowed and can be used for this assignment**
- **Data files**
  - **corpus.txt**: corpus containing a set of words and associated sentiment value
  - **stop_words.txt**: file containing a list of all stop words to delete for tweets
  - **unclassified_tweets.txt**: unclassified twitter data containing a set of tweets on Canadian election which needs to be analyzed for this assignment
  - **classified_tweets.txt** classified data containing a set of tweets which have been analyzed and scored for their sentiment

To Do:

1. Download both the classified and unclassified tweeter data, i.e, the *classified_tweets.txt* and *unclassified_tweets.txt* files.
2. Implement functionality to parse and clean a data file of the format (un)classified_tweets.txt. **(4.5 marks)**
   a. Write a function *clean_data(tw)* that takes in as input *tw*, a tweet string, cleans it by removing all punctuations and **returns** the cleaned tweet as output . (The function must have a **return** statement.) **(1.5 marks)**

```
def clean_data(tw):
    '''
    (str) -> str
    Input: a string tw (a tweet line)
    Output: a string whose content is that of tw with
     punctuations removed

     >>> clean_data("living the dream.#tommulcair
     instagram.com/p/8up9qepkxw/")
     'living the dream tommulcair instagramcomp8up9qepkxw'
     '''
```

   b. Write a function *remove_stop_words*(*tw*) that takes as input *tw,* a tweet string line, and **returns** the cleaned (stop words removed) version of the tweet as a string. Use the *stop_words.txt* file for this section. Note that before attempting to remove the stop words, all punctuations should be removed from the tweet. (The function must have a **return** statement.) **(1.5 marks)**

```
def remove_stop_words(tw):
    '''
    (str) -> str
    Input: a string tw (a tweet line)
    Output: a string whose content is tw with stop words removed
    >>> remove_stop_words("living the dream.#tommulcair
    instagram.com/p/8up9qepkxw/")
    'living dream tommulcair instagramcomp8up9qepkxw'
    '''
```

c. Write a function, *tokenize_unigram(tw),* that tokenizes a tweet *tw* into unigrams. For example, the tweet "Hello world!" would be tokenized into "hello" and "world". The tokens should not be stop words or contain punctuations. (The function must have a **return** statement.) **(1.5 marks)**

```
def tokenize_unigram(tw):
    '''
    (str)-> str
    Input: a string tw (a tweet line)
    Output: a list whose content which is broken into unigrams
     >>> tokenize_unigram("living the dream.#tommulcair
     instagram.com/p/8up9qepkxw/")
     ['living', 'dream', 'tommulcair', 'instagramcomp8up9qepkxw']
     '''
```

3. Write a function, *bag_of_words(tw),* that takes as input a tweet and creates a bag-of-words for it. A bag-of-words is a data structure that lists the number of times a word occurs in each tweet. When called on the tweet ***drink forgotten table drink***, *bag_of_words()* should return a Python dictionary {'drink': 2, 'forgotten': 1, 'table': 1}. A key in the dictionary is a word (token, as defined in 2.c) in the tweet and the associated value is that word's frequency in the tweet. The output of the function bag-of-words **MUST** be a **python dictionary** as described above. (The function must have a **return** statement.) **(2.5 marks)**

```
def  bag_of_words(tw):
      '''

      (str)->str

      Input: a string tweet of a tweet line ( a tweet line)

      Output: a python dictionary

      >>> bag_of_words("living the dream.#tommulcair

      instagram.com/p/8up9qepkxw/")

      {'living':1,'dream':1,'tommulcair':1, 'instagramcomp8up9qepkxw':1}
      '''
```

4. Write a function *party(tw)* that determines the political party for a tweet. *party*() takes as input a string (a tweet) and outputs a string, one of four possible categories, 'Liberal', 'Conservative', 'NDP' and 'Others'. In the docstrings of the function provide a description of the hashtags that you used to categorize the tweets into the four categories. For example, the hashtags ***#realchange, #justin*** *and* ***#trudeau*** correspond to tweets about the Liberal party. Write a

docstring paragraph explaining your rationality for hashtag usage. (The function must have a **return** statement.) **(2 mark)**

```
def party(tw):
    '''

    (str)->str

     Input: a string tweet of a tweet line ( a tweet line)

     Output: a python dictionary

     >>> party("living the dream.#tommulcair

     instagram.com/p/8up9qepkxw/")

     'NDP'
     '''
```

Steps 5 and 6 determine the sentiment of the tweets. To determine the sentiment of a tweet requires you to first develop an algorithm capable of calculating a score for each tweet. To assist you in developing this algorithm, use the *classifed_tweets.txt* file. The file *classified_tweets.txt* file contains a list of tweets which have already been analyzed. Take the tweets from the *classified_tweet.txt* file and develop an algorithm that outputs as close as possible to the scores given in the file. Once that is achieved, run that exact same algorithm against the unclassified tweets. **(6 marks in total)**

5. Write a function *tweet_score(tw)* to calculate a sentiment score for a tweet using the words it contains and their associated sentiment values. You can use *bag_of_words()* from step 3 to get a list of words in the tweet, and the data in corpus.csv file to get the sentiment values associated with some of them. Notice that not all words in a tweet will have associated pre-calculated sentiment values. It is up to you, how you calculate the overall score for a tweet. The score should be a number between 0 and 1, e.g., score of 0.8 would indicate a tweet that is more positive than negative. A tweet that your algorithm cannot classify using the data in the corpus should be given a score of -1. (**3 marks)**

```
def  tweet_score(tw):
      '''

      (str)->str

      Input: a string tweet of a tweet line ( a tweet line)

      Output: a floating point number between 0 and 1, or -1

      >>> bag_of_words("living the dream.#tommulcair

      instagram.com/p/8up9qepkxw/")

      0.8
      '''
```

6. Write a function *tweet_classifier(tw)* that calls *tweet_score()* and outputs a tweet's sentiment as a binary value, 0 or 1, or leaves the score at -1 if the tweet cannot be classified by your algorithm. It is up to you to test your function *tweet_classifier()* on the classified data to design an algorithm for converting continuous sentiment scores from the interval [0, 1] into sentiment binary value. **(3 marks)**

```
def tweet_classifier(tw):
      '''
```

```
(str)->str

Input: a string tweet of a tweet line ( a tweet line)

Output: an integer, either 0 or 1 or -1

>>> tweet_classifier ('living the dream.#tommulcair

instagram.com/p/8up9qepkxw/')

1
'''
```

What to submit:

Submit via Blackboard a Python .py file containing your implementation of the functions in sections 2 to 6 with the following naming convention

studentnumber_lastname_assignment1.py

Make sure that you comment your code appropriately and describe your algorithms in sufficient detail. Your module should be self contained, i.e., the functions you submit cannot call functions you defined in other Python modules. Calls to *clean_data, remove_stop_words*, *tokenize_unigram, bag_of_words, party*, *tweet_score* are permitted (you can use some of the functions you implement for the first parts of the assignment in the functions you implement for the latter parts).

Note: DO NOT place any print() or input() statements in the functions you submit.