**CSC 202: Intro to Machine Intelligence**
**Natural Language Processing Project**

You must turn in a Jupyter Notebook that carries out the appropriate analysis. Your notebook should begin with your name and a one paragraph summary describing the main goals of your project (e.g., to find the most common words in a collection of tweets). Each code cell should be preceded by a markdown cell that describes what the code does (as in the class notes). Your Notebook should be correctly formatted with an appropriate title, section headers, and subsection headers. [20 points]

1. Complete **one** of the following to collect data to analyze [25 points]:

- Collect at least 200 tweets, using pagination to get multiple pages of results (see the second example at https://docs.tweepy.org/en/latest/v2_pagination.html)

- Retrieve data from at least one Wikipedia page, but the more the better (see previous lab for an example)

- Read data from a plain text file, which can be done using the code below (the file should be a plain text file (with a .txt extension) from the same directory as your Notebook):

```
o  with open('file.txt', 'r') as f :  # open a file for reading
       text = f.read() # stores the text of the file in text
```

  For possible data sets, the following web sites may be useful. Note that you will need to copy or save the text to a plain text file.

  - Public domain books: Project Gutenberg (https://www.gutenberg.org/)
  - Song lyrics: https://www.lyrics.com/
  - Movie scripts:
    - The Internet Movie Script Database, https://imsdb.com/
    - Awesome Film, http://www.awesomefilm.com/

2. Complete any **three** of the following [25 points each]

   a. Generate a table showing the 20 top most frequently occurring words and their frequencies, and generate a wordcloud of the text. For this analysis, stopwords should be removed, and all words should be converted to lowercase. You may also choose to *stem* or your text, and may choose to restrict the wordcloud to, e.g., the top 50 words if it is too cluttered.

   b. Repeat the above using *n-grams*. An *n-gram* corresponds to a set of *n* consecutive words. A *bigram* is an *n-gram* where *n = 2* and a *trigram* is an *n-gram* where *n = 3*. For example, the sentence "how are you?" has bigrams of "how are" and "are you". Using *textblob*, *n-grams* can be accessed using *blob.ngrams(n)*. Generate a table and/or wordcloud showing the most common bigrams and/or trigrams.

c. Sentiment analysis (select at least one)

   i. Output the five most positive sentences, lines, or tweets, and the five most negative sentences, or lines, or tweets. This requires creating a list of sentence-polarity pairs, and then sorting the results by polarity. The sorting process is very similar to sorting word-frequency pairs by frequency.

   ii. Compare the sentiment of two different text collections (e.g., tweets about two different topics, or two different chapters). Recommended: compare sentiments visually using a boxplot. An example of a boxplot can be shown here: https://i0.wp.com/datavizpyr.com/wp-content/uploads/2020/06/customize_mean_mark_in_boxplot_Seaborn_boxplot_Python.png?resize=605%2C484&ssl=1

   In order to generate a boxplot, create a data frame where each column contains the sentiment values for a different group. The code below uses *pd.concat* to concatenate (combine) data frames, which will handle the case where the groups do not have the same number of sentiment values.

```
import pandas as pd
import seaborn as sns

# lists contains the sentiments for each group
group1 = [-.5, 0, .2, -.4]
group2 = [0, .3, .8, -.1, .3]

# create data frames for each group
df1 = pd.DataFrame({'group1':group1})
df2 = pd.DataFrame({'group2':group2})

# combine the data frames
df = pd.concat([df1,df2], axis=1)

# generate the boxplot
sns.boxplot(data = df)
```

d. Use named entity recognition to output the most commonly mentioned entities for various types, such as the most common people (PERSON), organizations (ORG) or geopolitical entities (GPE). Hint: use a *Counter* object to count the frequency of each element in a list (for an example see https://www.hackerrank.com/challenges/collections-counter/problem)

e. Other analyses may be acceptable. I encourage you to  be creative and contact me with ideas.