CSC 202: Intro to Machine Intelligence
Natural Language Processing Project

In our unit involving Natural Language Processing, we covered how to analyze text in Python using the *TextBlob* and *Spacy* modules, how to mine Twitter data, and how to develop a digital assistant using IBM Watson. In this project, you will either (1) use natural language processing for text-mining to analyze one or more texts, or (2) develop a digital assistant.

Option #1. Text mining

Analyze one or more sets of texts by completing the requirements below. Example project ideas include

- Analyzing all tweets from a specific user by looking at their timeline
- Analyzing tweets containing a certain keyword or trending topic
- Comparing tweets across different users or across different topics
- Analyzing a text, such as an e-book, Wikipedia entry, news article, or a political speech

If choosing this option, 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 1000 tweets (or 200 tweets if retrieving tweets for a single user) using a Tweepy cursor to get tweets across multiple pages (for examples, see the following page: http://docs.tweepy.org/en/v3.8.0/cursor_tutorial.html)

- Retrieve data from at least one Wikipedia page (see Lab 8 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 from the same directory as your Notebook:

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

  For possible data sets, Project Gutenberg (https://www.gutenberg.org/) contains the text of many public domain books.

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

a. Generate a table showing the top 20 words and generate a wordcloud Stopwords should be removed, and all words should be converted to lowercase. You may also choose to *stem* or *lemmative* 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.ngram(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 (or tweets), and the five most negative sentences (or tweets).

    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 pandas boxplot (https://wellsr.com/python/python-create-pandas-boxplots-with-dataframes/)

d. From a collection of tweets, extract the location of each user. Display a map with markers at each location containing the text of the tweet (show at least 5 markers).

e. 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)

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

Option #2. Digital Assistant

Develop an IBM Watson Assistant and optionally demonstrate it using Python and the IBM cloud API. For example, an assistant can be developed that

- Allows the user to place a food order
- Provides the user with information, such as
  - Movie information, including description, ratings, and showtimes
  - Song information, including artists, songs, albums, and release dates
  - Trivia, such as states and their capitols

If choosing this option, you will turn in a the JSON file exported from the Assistant, and any additional Python code (in a Jupyter Notebook), if applicable.

1. Create a Watson Assistant that correctly carries out a task. Your assistant does not need to cover all possibilities but should work correctly for the entities included. For example, an assistant that provides information about Eastern Faculty should include at least 5 faculty members, and should work correctly for the intents included. If the user wants information about another faculty member, or requests information (has an intent) that is not available, the assistant should respond appropriately. Your assistant should include

   a. At least 5 intents with at least 5 user examples [20 points]

   b. At least 5 entities with appropriate values and synonyms. At least one entity must have 5 values. [20 points]

   c. At least 3 context variables that are properly set and utilized. Recall that a context variable allows the assistant to remember what the user has said (for example, the faculty member the user is interested in) [20 points]

   d. A dialogue that uses at least one system entity such as @sys-date or @sys-time (https://cloud.ibm.com/docs/services/assistant?topic=assistant-system-entities) or a *pattern entity* that recognize patterns such as an e-mail address or phone number (https://cloud.ibm.com/docs/services/assistant-icp?topic=assistant-private-entities) [10 points]

   e. A dialogue that correctly converses with the user [30 points]

   f. *Optional*: Use the IBM cloud API to provide a demo for the assistant (https://cloud.ibm.com/apidocs/assistant/assistant-v2?code=python). Results are returned in JSON format, which can be converted to a dictionary. Use Python's SpeechRecognition module so a user can "talk" the assistant using the computer's microphone.