The purpose of this assignment is to test whether you meet the basic prerequisite for this class. You must complete the assignment independently, without getting help from other people. This assignment must be submitted to canvas by **Jan 18 11:59 PM Phoenix Time**. If you are enrolled into the class after Jan 16, please submit your assignment **within 2 days** (48 hours) after you are enrolled.

Q1: Data Processing

In order to train machine learning algorithms on raw text documents, we first need to represent them in a way that the machine can process. One way is by computing a Term Frequency Inverse Document Frequency (TFIDF) vector for each document. Terms with a high TFIDF score are often those that best characterize the topic of each document. There are multiple ways to define TFIDF; in this assignment we will use the following definition:

$$\text{TFIDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_j$$

Given a collection of *n* documents, let f_{ij} be the frequency (number of occurrences) of the word, *j*, in document, *i*. The term frequency, TF_{ij} is defined as:

$$TF_{ij} = \frac{f_{ij}}{max_k f_{ki}}$$

where the term frequency of word *j*, is normalized by the maximum number of occurrences of any word, *k*, in that same document. Therefore, the most frequent term in document, *i*, gets a *TF*=1. To compute the inverse document frequency, let m_j be the number of documents in which word *j* appears. Then IDF_j is:

$$IDF_j = log_{10}(\frac{n}{m_j})$$

Instructions:

- Download dataset here:
  - The dataset contains raw text of 1000 news articles and the article category. Each row is a document.
  - The raw file is a .csv with three columns: `ArticleId, Text, Category`
  - There are five categories in the dataset:
    - sport
    - business
    - politics
    - entertainment
    - tech
- Generate unigrams for each document with sample code:
  - Turn the document into lowercase
  - Remove punctuation
  - Tokenize document
  - Remove stop words
  - extract character stems
- After those steps, you will get a list of words, which can be considered as the unigram. We will only consider words in a predefined dictionary: dictionary.txt . There are 1,000 unigrams in the dictionary. Words not in this list should be ignored.

- Once each document is represented as a set of unigrams and you have filtered the set using dictionary.txt, compute the TFIDF matrix for the corpus. **Round your answer to 4 decimal places.** You need to implement TFIDF calculation according to the description above. Do NOT use any existing TFIDF calculation function
- Given 1000 documents and 1000 words you should output a (1000 x 1000) matrix, where element *ij* corresponds to the $TFIDF_{ij}$ score for document *i* and word *j*. The order of the 1000 columns should correspond to the order of words in the provided dictionary. Additionally, the order of the documents should be the same as the order in the raw text file (24_train_3.csv).
- Compute the top 3 most frequent words in each category
- Compute the top 3 highest **average** TFIDF words in each category

Deliverables:

Please submit a single .zip file called "`{your_name}`_data_processing.zip", with the following directories. Please make sure that you do not zip a folder (which would highly likely happen if you firstly put all the files in a folder and zip the folder), but you should directly zip the files together in a zip file. Simply test by running ``unzip `{your_name}`_data_processing.zip'' on your shell and you will get a folder in the following structure.

Folder directory:

`{your_name}`_data_processing

|---- .ipynb

|---- matrix.txt

|---- frequency.json

|---- scores.json

This homework has 100 pts in total and will take 5%. All the file names and formats are strictly reinforced. Any mismatch with the required format will result in zero score for certain questions.

- A .ipynb that includes all your codes (10 pts). Python is required.
- A file with the processed *TFIDF* matrix, called `matrix.txt` (30 pts). For this file, each line will be have 1000, TFIDF scores that represent a document, split by commas and no space between numbers (e.g, 5 numbers that split by commas look like this: 1,2,3,4,5) . There will be 1000 rows in total, one for each document. Please encode the file using Unicode (not in matrix format in matlab or any other software).
- A json file with the top 3 most frequent words by category, called `frequency.json` (see example below). (30 pts)
- A json file with the top 3 highest **average** TFIDF words by category, called `scores.json` (see example below). (30 pts)

Example JSON submission:

Make sure you parse in a well-formatted JSON - check online here.

You are to output a JSON for `frequency.json` and `scores.json`. Each category (e.g., sport, business) will be a key. The value for each will be a dictionary mapping words (e.g., "baseball") to the score (e.g., .9538). Use the word count as the score in `frequency.json` and the TFIDF as the score in `scores.json`.


```
{
    "sport": {
                "baseball": 0.9512,
                "ball": 0.9333,
                "stadium": 0.8837
            },
    "business": {
                "ceo": 0.9529,
                "company": 0.9368,
                "sales": 0.8892
            }
}
```