INFO 153 HW5

This assignment is to process a set of text files and compute related TF and IDF statistics.

## Data Files

Please collect about ***20 text data instances*** (e.g. brief news reports or research abstracts) and save them as individual .txt files. Files names should be named in a sequential order such as 1.txt, 2.txt, 3.txt, … and 20.txt.

## Create a *Document* abstract data type/class

The Document class should have:

* A dictionary variable to keep track of all unique words and their frequency in the document;
* A tokenize(text) method that:
  + Splits text into single words using space and punctuation as delimiter;
  + Use a loop to go through all the words, and for each word:
    - If it does not appear in the dictionary, add it to the dictionary and set its count/frquency to 1;
    - If it is already in the dictionary, increment its count/frequency by adding 1 to it;

## Create a save\_dictionary function

The function should accept two arguments:

* One argument for the dictionary with data to be saved;
* Second argument about the file pathname to save the data;

The function saves all data/statistics in the dictionary to text files, with each key-value pair in one text line separated by a tab (“\t”). The output file should look like:

*Key1 value1*

*Key2 value2*

*Key3 value3*

*…*

## Create a *vectorize* function

The function should:

* Take a string argument as the path to where the text data files are;
* Process all data files in the path and produces TF and IDF statistics;

Here are steps in the function:

* Create a dictionary variable to keep track all unique words and their DF (document frequency);
* List all .txt files in the path argument;
* For each file:
  + Create a Document object (based on the Document class);
  + Read the content (text lines) from the text file;
  + Call the document object’s tokenize function to process the text content;
  + Call the save\_dictionary function to save the document’s dictionary with TF (term frequencies) to a file, where the filename should be **tf\_DOCID.txt** in the same path.
    - For example, after processing 1.txt file, the data should be saved to tf\_1.txt file in the same directory.
  + **Create a nested loop, and for each word in the document’s dictionary:** 
    - **If it does not appear in the dictionary for DF, then add the word to the DF dictionary;**
    - **If it is already in the DF dictionary, increment its DF value by adding 1 to itself;**
* After all files are processed, call the save\_dictionary function again to save the DF dictionary to a file named **df.txt** in the same path with the input text files.

## Bonus:

Compute pair-wise cosine similarities among the documents using Term Frequency (TF) weights, and save the results to a file.