Final Project / MS1 and MS2
Word Processor

(Out of 100 marks, 20% of your final grade)

I. Objective

In this project, you will code and execute python modules that contain functions useful for indexing and searching into a collection of documents. **The project consists of two Milestones (MS1 and MS2)**.

II. Learning Outcomes

Upon successful completion of this assignment, you will have demonstrated the abilities to implement and call functions as follows:

1. Create the main menu for selecting options for user input and a jump table for functions.
2. Manipulate lists and dictionaries to create an index of a whole collection of documents.
3. Implement a user query text preprocessing
4. Implement query-document matching model to retrieve relevant documents

III. Marking of Submission

Marking scheme:

- **Milestone 1 (80%)**
- **Milestone 2 (10%)**
- **Report (10%)**

Due date:

- The due date for MS1: Wednesday, April 6th, 8:00 AM
- The due date for MS2(entire project): Wednesday, April 13th, 8:00 AM

Late submissions:

- 1 Day delay submissions: 10% deduction
- 2 Days delay submissions: 20% deduction
- 3 Days delay submissions: 30% deduction
- 4 Days delay submissions: Not accepted.

Evaluation Criteria:

Grading is based on:

1. Delivered report and overall match with others and internet
2. One-on-One discussion and/or presentation. Good to have power points.
3. **-% for late submission/day**
4. Following the recommended format in writing the report and functions, including references and questions/answers
5. Running and understanding the code.

IV. Instructions

- **Docstring Part**

  Fill in the docstring part at the beginning of the python module and also for each function, by describing of the input, program functionality, and output. As discussed in our lectures

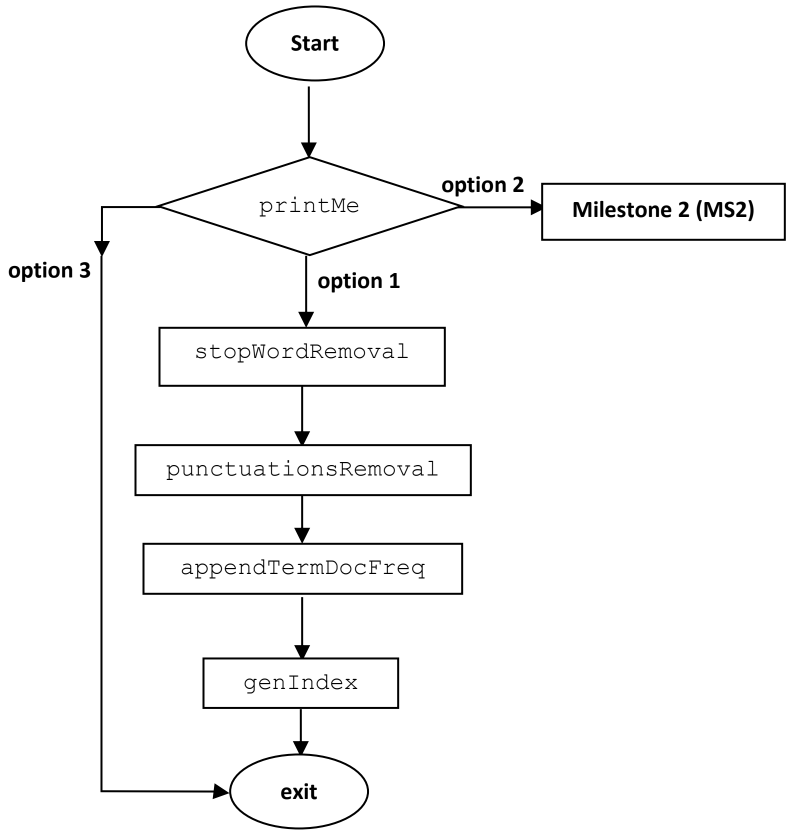
- **User input**

  A collection of text documents with `.txt` extension. This collection can be found in the folder dataset of the Project folder on the Blackboard.

V. MS1: The indexing Module [80%]

The indexing module aims at generating an index for a collection of documents. It starts by reading a collection of text documents, processing each of the document's content, and extracting its terms and their frequencies. After processing all documents, a dictionary that serves as an index is created that goes from terms to a list of documents containing the terms and their frequencies in each of the documents containing the term. See below flowchart.

Note: use MS1.py provided



The following functions should be implemented:

- **`def printMenu():`**
  This function displays the following menu to the user. The user must select 1 for indexing and 3 to exit. This function checks for valid/invalid input. If invalid input is entered, the function prompts the user again and prints an informative message. If the input is valid, it is returned. **Note: you will upgrade this function in MS2.**

  ```
  Menu:
  Please enter 1 for indexing and 3 to exit
    1. Indexing
    3. Exit
  ```

- **`def readFolderContent():`**
  The code for this function is given. This function reads all the text files in the folder **`dataset`** and appends them to a list. Each item in the list is the content of a text file in the dataset.

- **`def indexing():`**
  The indexing module aims at generating an index for a collection of documents. It starts by reading a collection of text documents, processing each of the document's content, and extracting its terms and their frequencies. After processing all documents, a

dictionary that serves as an index is created that goes from terms to a list of documents containing the terms and their frequencies in each of the documents containing the term. See the flowchart above for more information.

- **`def stopWordRemoval(text):`**
  This function takes a text as an argument, removes all the stop words from the text, and returns the text. A list of stop words is provided for you on the Blackboard/Project folder. For example, words `"the"` and `"on"` are considered as stop words. Thus, they are removed from the following input.
  input: `"The monkeys jump on the bed."`
  output: `"monkeys jump bed."`

- **`def punctuationRemoval(text):`**
  This function takes a text as an argument, removes all the punctuations from the text, and returns the text. Here is a list of punctuations you may use in your code.

  ```
  punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
  ```

- **`def appendTermDocFreq(cleanText, termDocFreqFile):`**
  This function takes a clean text as an argument and appends the file **`TermDocFreqFile`** with the list of terms, the document in which they appear, and their frequencies. **`termDocFreqFile`** format: 3 columns, values separated with space
  output:

  ```
          Term doc# freq
          recipe 1 5
          sweet 1 1
          sugar 1 2
  ```

- **`def genIndex(termDocFreqFile):`**
  This function reads **`termDocFreqFile`** line by line and appends the global index that goes from terms as keys to the list of documents that contains them with their frequencies. Appending the index works as follows:
  for line in **`termDocFreqFile`**,
  - if the term does not exist in index, add the term as key, the value will be a dictionary **`containg docid:freq`** as **`key:val`** in index
  - if the term already exists in the index, append the **`val`** (which is a dictionary) with **`docid:freq`**

VI. MS2: The retrieval module [10%]

Completion of this part is not a requirement to pass the course. However, you are encouraged to complete it to get the complete score for this project.

The retrieval module aims at searching into a collection of text documents. It mainly prompts the user for a text query, and search for documents that contain the query terms. Each document is associated with a score based on TF*IDF. Documents are then ranked based on their relevance score. The following functions should be implemented:

- **`def search(query)`**: This function preprocesses the query text and splits it into individual terms `(w)`. Then find the documents containing the terms and rank them by relevance score. The function returns a dictionary of documents and their scores.
- The relevance of a document **D** with respect to query **Q** is calculated as follows:

$$s(Q, D) = \sum_{w} tf_{w,D} . \log \frac{|C|}{df_w}$$

- $tf_{w,D}$ is the frequency of term w in document D, $|C|$ is the total number of documents in the collection, df_w is the number of documents that contain term w. module aims at searching. Here is the pseudocode:

Pseudocode:

```
- S(q,D) = 0
- Then, for each term (w),
- Calculate $tf_{w,D}$ : That is the frequency of term w in document D. You
  may use the termDocFreqFile generated in milestone 1 to find this
  value.
- Calculate $df_w$: That is the number of documents that contain term w.
  You may use the termDocFreqFile generated in MS1 to find this value.
  The length of dictionary for each word.
- Calculate $tf_{w,D} . \log \frac{|C|}{df_w}$
- $S(q, D) = S(q, D) + tf_{w,D} . \log \frac{|C|}{df_w}$
```

Your program calculates the relevance score $S(q, D)$ for a given query for all the documents and returns the document that has largest relevance score as the most relevant document.

More details: https://www.youtube.com/watch?v=nHnML6fauDg

The main module (MS1 and MS2)

As shown in the above flowchart, implement the main function that shows a menu with the following options:

1. Indexing
2. Retrieval
3. Exit

When the user selects **option 1**,

- The system prompts the user for the path to the directory of text documents to index
- Call **`runindexing()`** function which will result in filling up the index.

When the user selects **option 2**,

- The system will prompt the user to enter a query text.

- Call **`search()`** function to find the list of relevant documents and their scores.

Display the list of documents by their ids and their scores.

VII. Report **[10%]**

1. Write a paragraph about what you have learned in this project, and what you found most challenging.
2. Write a paragraph about some of the applications of document indexing and text processing.
3. Why is it a good idea to remove stop words and punctuations?
4. In document retrieval, why do we use TF*IDF method? Why using only term frequency is not a good idea?

VIII. SUBMISSION

Submit your word, pdf and ipynb files on BB. There will be a demo for this assignment.