Homework Assignment 1: Due 09/22/2024 at 11:59 PM

You may use whatever IDEs Platforms / text editors you like, but you must submit your responses and source code files (Jupyter notebooks are considered as such) on iCollege.

Note: Please refer to the Jupyter notebooks created in class. They provide useful hints for solving the following problems. In the submitted Jupyter notebook(s), your code cells' output (if any) should be visible **without** executing the notebook.

1) (27 points) Download the resource file "sw-security-urls.txt". Create a Jupyter notebook (in Python) that performs the following tasks:
   a) (4 points) Write and call a function that downloads the contents of all URLs (please note that the URLs provided are in UTF-8 encoding) in the resource file (Wikipedia articles on software security) one by one to a local directory. Afterwards, this local directory should contain 85 HTML-files.
   b) (9 points) Using the library BeautifulSoup, extract from each downloaded HTML-file its title, its main text found under the element <div id="bodyContent" …>, and the following JSON-LD formatted metadata (if existing): "name", "url", "datePublished", "dateModified", "headline". Store this data for each file in a dictionary and add this dictionary to a list. This list should then contain 85 dictionaries. Output this list.
   c) (3 points) At the same time, create a string that contains the main text of all the Wikipedia articles. To do that, you can concatenate the individual main texts extracted in the previous step (leave a whitespace between each text). Finally, save the content of this string in a local text file.
   d) (6 points) Create a pandas DataFrame (e.g. named "articles") from the list obtained in step 1b) and set its index to the article's title. Output this DataFrame. Which Wikipedia articles do not provide "dateModified" information in their JSON-LD formatted metadata? (You can use a Documentation cell in your Jupyter notebook to provide your answer.)
   e) (3 points) Sort the DataFrame based on the length of the 'headline' column in ascending order, assign this sorted DataFrame to another DataFrame. Then select the first 5 rows from this DataFrame. Output this selection.
   f) (2 points) Store the obtained "articles" pandas DataFrame (set index_label to "title") in a local sqlite3 database.

2) (16 points) Extend the Jupyter notebook (in Python) created in 1) by providing solutions to the following tasks:
   a) (4 points) Create an NLTK-corpus using PlaintextCorpusReader from the local text file created in 1c). For the segmentation of sentences, please use the RegexpTokenizer with the pattern: r'[^.!?]+' . Using NLTK's method words(), output at most 200 words of the corpus together with their absolute frequency in descending order (by frequency). Note: this output might contain elements like punctuation marks and stop words.
   b) (7 points) Filter out the following elements from the word list obtained in the previous step:
      - any stop words found in NLTK's English stop word corpus
      - any elements that contain only one character
      - any elements that do not contain the letters A-Z and a-z (Regular expression pattern: r'[^A-Za-z]')

Also, make sure that the filtered word list contains only lowercase words. Output this filtered list of (at most 200) words along with their absolute frequency in descending order (by frequency). To achieve this, count the occurrences of each word using a defaultdict after the filtering is complete.

c) (5 points) Create and output a word cloud using the defaultdict obtained in 2b).

3) (17 points) Extend the Jupyter notebook (in Python) created in 1) and 2) by providing solutions to the following tasks:
   a) (1 point) Prove that the database file from 1f) has been properly loaded into a pandas DataFrame by showing (output requested) that the saved table exists and the pandas DataFrame contains the expected columns “title” (as index), “text”, “name”, “url”, “datePublished”, “dateModified”, “headline” as well as 85 rows for the respective Wikipedia texts/documents.
   b) (2 points) Copy the three functions “tokenize()”, “remove_stop()”, and “prepare()” from the Jupyter notebook “Textbook_Blueprints_TFIDF.ipynb” and create a simple pipeline (a list) consisting of the entries “str.lower”, “tokenize”, and “remove_stop”. Modify the copied function “tokenize()” such that it returns “re.findall(r'[A-Za-z]+', text)”.
   c) (1 point) Apply the function “prepare()” to the column “text” of the DataFrame and add a new column called “tokens” to it.
   d) (1 point) Apply the function “len()” to the column “tokens” of the DataFrame and add a new column called “num_tokens” to it.
   e) (1 point) Show that the DataFrame actually contains the two newly added columns by outputting two random rows.
   f) (2 points) Copy the function “count_words()” from the provided Jupyter notebook “Textbook_Blueprints_TFIDF.ipynb” and create a new DataFrame “freq_df” that is returned by calling “count_words()”. (Of course, “count_words()” must operate on the DataFrame loaded in 3a).) Output the first 10 rows of “freq_df”.
   g) (2 points) Copy the function “compute_idf()” from the Jupyter notebook “Textbook_Blueprints_TFIDF.ipynb” and create a new DataFrame “idf_df” that is returned by calling “compute_idf()”. (Of course, “compute_idf()” must operate on the DataFrame loaded in 3a).)
   h) (1 point) Perform a left join on “freq_df” with “idf_df” using the available “join()” method. Assign the resulting DataFrame, which completely incorporates the columns from both DataFrames, to the variable “freq_df”. Be aware that “freq_df” will be updated to the new DataFrame as a result of this operation.
   i) (2 points) Calculate the term frequency-inverse document frequency (TF-IDF) values by multiplying the corresponding values in the “freq” and “idf” columns of the DataFrame “freq_df” and assign the resulting values to a new column called “tfidf” in “freq_df”.
   j) (1 point) Sort the DataFrame “freq_df” based on the values in the “tfidf” column in descending order and output the first 10 rows from this sorted DataFrame.
   k) (3 points) Are these returned rows more meaningful than the first 10 entries of the filtered list obtained from 2b)? Please provide a brief explanation. (You can use a Documentation cell in your Jupyter notebook to provide your answers.)

Good luck!