Homework Assignment 2: Due 10/06/2024 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) (40 points) Download the resource files "wiki_articles_hw1.db" (it is the sqlite3 database file created in the previous homework) and the Jupyter notebook "hw2_skeleton.ipynb". Extend this notebook by solving the following tasks in the Python language:
   a) (1 point) Prove that the database file has been properly loaded by showing (output requested) that the table "wiki_articles_hw1" exists and the pandas DataFrame df contains the expected columns "title", "text", "name", "url", "datePublished", and "headline" as well as 85 rows for the respective Wikipedia texts/documents.
   b) (3 points) Write a function "pos_tag(d)" that extracts the following elements for each token in a given spaCy document d: "token.text", "token.pos_", "token.tag_", "token.lemma_". Furthermore, create a pandas DataFrame with this data and return this DataFrame.
   c) (3 points) Write a function "noun_chunks(d)" that extracts the following elements for each token in a given spaCy document d: "chunk.text". Furthermore, create a pandas DataFrame with this data and return this DataFrame.
   d) (18 points) For each text in df["text"] (contains each Wikipedia article's main content)
      (1) (4 points) Remove the lines that start with "From Wikipedia, the free encyclopedia". Also, delete all footer content starting with "Retrieved from…", which includes lines that start with "Category:…" or "Categories:…". Then, remove all (typically) Wikipedia-related square brackets along with their contents. Thus, elements like "[5]" or "[edit]" should be removed. Lastly, replace all occurrences of "\n" with a space (" ").
      (2) (1 point) Create a spaCy document from it using spaCy's pipeline.
      (3) (6 points) Extract the noun chunks using the function in 1c). For this purpose, extract only those noun chunks that contain at least 1 space. Also, remove possibly existing leading and trailing whitespaces from them and convert them to lowercase. For each extracted noun chunk this way, count in a defaultdict how often it occurred (cumulated over all texts).
      (4) (7 points) Extract the lemmas using the function in 1b). For this purpose, extract only those lemmas for tokens that are assigned the POS-tags "NOUN" or "PROPN". Also, only regard those lemmas that are longer than 1 character and convert them to lowercase. Finally, create a reverse word index using another defaultdict, in which you store for each lemma in which text it occurred. The defaultdict should therefore map from a lemma (the key; you can use the actual string of the lemma) to a list of numbers that represent the respective texts (you can assign this text/document number while looping through all texts) which contain the key. You do not need to count the frequency of lemmas.
   e) (2 points) Output the 10 most frequent noun chunks extracted in 1d(3) along with their absolute frequency using the function most_common() of the Counter class.
   f) (1 point) Output the number of keys in the reverse word index created in 1d(4).

g) (2 points) Output a list of those lemmas in the reverse word index that appear in at least 50 texts/documents.
h) (2 points) How many unique lemmas occur in only one of the 85 documents?
i) (4 points) The list returned by the Counter-function most_common() in task 1e) is ordered in descending order and thus ranked. The token/word/noun chunk at position (rank) 0 occurs most often. Create a line chart (X-axis for rank, Y-axis for frequency) using the Matplotlib library based on the return of the function most_common() for the 1000 most frequent noun chunks. Label the axes accordingly.
j) (4 points) Create another line plot under the same conditions as in the previous task, but this time as a log-log plot. For this purpose, please also refer to the documentations at https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.loglog.html , https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xscale.html , and https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.yscale.html .

2) (10 points) In downstream analytical steps, such as co-occurrence analysis, the inclusion of rare terms can negatively impact the determination of the significance of co-occurrences. Explain the reasons for this and suggest potential solutions. Additionally, discuss how misleading significance values can affect subsequent tasks. Also, explain why these solutions are sensible. Please provide your answer in the Jupyter notebook for task 1).

Good luck!