Spring 2023 – 8980 NLP

**Homework Assignment 3**: Due 03/05/2023 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) (21 points) Download the resource files "wiki-articles-extended.db" (it is the sqlite3 database file created in lecture 10) and the Jupyter notebook "hw3_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_extended" exists and the pandas dataframe "df" contains the expected columns "title", "text", "name", "url", and "nouns" (along with many other columns) as well as 60 rows for the respective Wikipedia texts/documents.
   b) (2 points) Extract the most common 7500 nouns from all the 60 Wikipedia texts. These tokens must be in lower case and have a length of at least 1. Also, they cannot be stop words. For this purpose, please use the code fragments provided. Afterwards, the object "nouns" will map from a token to its overall frequency. The list "voc" contains the strings of tokens extracted, sorted in descending order of their frequency of occurrence (the token at the largest index position 7499 will be the least frequent one). Thus, this list can be used to find an index for a specific token. For instance, the index of token "information" in this list could be 3, found by calling *voc.index("information")* . Show that the list "voc" contains 7500 tokens and that there are no tokens with a frequency of 0 in it (the last element in it must appear at least 1 time).

   c) (15 points) Create a co-occurrence matrix (named e.g. "coocc_matrix") using a 2-dimensional Numpy array that has a shape of (7500x7500). Each element in this matrix should be initialized to 0 and will count how often a token i appears together with a token j in df["nouns"] using a co-occurrence window of 5. The tokens to be extracted (in lower case, no stop words) from df["nouns"] must appear in the previously obtained list "voc". For instance, the element at (3, 0) could be accessed by "coocc_matrix[0][3]". As an undirected co-occurrence graph is generated this way, the value of "coocc_matrix[0][3]" must match that of "coocc_matrix[3][0]". The mentioned indices 0 and 3 can be obtained by calling *voc.index("information")* and *voc.index("security").*

      Note: Co-occurrences of a token with itself in the co-occurrence window should be counted.
   d) (3 points) For all indices i and j, prove that the elements "coocc_matrix[i][j]" and "coocc_matrix[j][i]" have the same values.

2) (7 points) Extend the Jupyter notebook (in Python) created in 1) by providing solutions to the following tasks:
   a) Using the basic PageRank algorithm (not the extended variant) applied to a stochastic matrix derived from the co-occurrence matrix obtained in 1c), output the ten most important keywords/tokens of "voc" sorted in descending order by their PageRank values.

      Note: For this purpose, you can use the function "pagerank()" from the Python-example under https://en.wikipedia.org/wiki/PageRank , which computes the PageRank scores using the power method described in this Wikipedia-article. In order to be able to do so, you must first transform the obtained co-occurrence matrix into a stochastic matrix by e.g. calling *stochastic_matrix = coocc_matrix / (coocc_matrix.sum(axis=0, keepdims=True)+0.001)* . Then, the mentioned function "pagerank" can be called by *v = pagerank(stochastic_matrix)* .

3) (22 points) Extend the Jupyter notebook (in Python) created in 1) and 2) by providing solutions to the following tasks:
   a) (5 points) Create a copy (named e.g. "dice_matrix") of the co-occurrence matrix obtained in 1c), calculate the significances of the co-occurrences using the Dice coefficient, and update the elements in "dice_matrix" accordingly.
   b) (2 points) Output the Dice significance values of the co-occurrences ("software", "security") and ("security", "software"). These must match.
   c) (5 points) Now calculate the value of the cosine similarity between the context vectors of the tokens "software" and "security". You could do that by relying on Numpy's functions "np.dot()" and "np.linalg.norm()" or by making use of the function "cosine_similarity()" from "sklearn.metrics.pairwise". In the latter case, the vectors may have to be reshaped. Note: The context vector of a term i can be obtained by accessing the i-th row in "dice_matrix". For instance, the context vector of the term "software" can be accessed by calling *dice_matrix[voc.index("software")]* .
   d) (3 points) Compare the values obtained in 3b) and 3c). What do you notice? How do you interpret the different values? Formulate complete answer sentences and enter them in the Jupyter notebook.
   e) (7 points) What are the 10 most similar words/tokens to "software" in descending order of their cosine similarity scores? For this purpose, calculate the cosine similarity values between the context vector of the token "software" and all row vectors in "dice_matrix". Hint: The function "cosine_similarity()" from "sklearn.metrics.pairwise" can be passed this matrix as a single parameter. Note: the most similar token to "software" must be "software".