Homework Assignment 4: Due 11/03/2024 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) (18 points) Download the resource files "wiki_articles_hw1_extended.db", "wiki_articles_hw1_extended.w2v", and the Jupyter notebook "hw4_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_extended" exists and the pandas DataFrame "df" contains the expected columns "title", "text", "name", "url", and "nav" (along with many other columns) as well as 85 rows for the respective Wikipedia articles on the topical field of software security.
   b) (2 points) Using the provided code fragments, create the document-term matrix with the "TfidfVectorizer" applied to the DataFrame's column "nav", (optionally) train a Word2Vec model using this column, and load the thus obtained word vectors. Note: The resource file "wiki_articles_hw1_extended.w2v" already provides suitable, pre-trained word vectors. However, feel free to create your own ones.
   c) (15 points) Semantic Transformation: Design and implement an approach to represent each of the 85 documents by exactly one real-valued embedding vector (a so-called document embedding) using the word vectors and TF-IDF weights obtained in 1b). The output should be a list of document embeddings generated this way. Note: You are not allowed to use existing approaches to create document embeddings such as Doc2Vec. Your approach to solve this task could be inspired by the method of calculating text-representing terms (TRCs) in co-occurrence graphs, which has been discussed in class. However, in this setting, we are not working with co-occurrence graphs, but with word embeddings instead.

2) (7 points) Extend the Jupyter notebook (in Python) created in 1) by providing solutions to the following tasks:
   a) (4 points) Using the word vectors from the resource file "wiki_articles_hw1_extended.w2v" and the document embeddings obtained from 1c) determine the document most similar to the token "scareware" from the set of all 85 analyzed Wikipedia articles by means of the cosine similarity measure and print out its title.
   b) (3 points) In your opinion, is the result of 2a) plausible? Explain briefly why you think so.

3) (20 points) Extend the Jupyter notebook (in Python) created in 1) and 2) by providing solutions to the following tasks:
   a) (6 points) Apply the k-means cluster algorithm on the document embeddings obtained from 1c) while setting the number k of expected clusters to 10. For all clusters, list the titles of the documents contained in them.
   b) (8 points) For all clusters obtained from 3a), determine the mean similarity of all document pairs (if those exist) contained in them and print out this value. For this purpose, use the cosine similarity measure.

c) (6 points) Interpret the similarity values obtained in 3b). In doing so, also refer to the general topical field of the Wikipedia corpus used. Does the clustering result confirm the result obtained in 2a)?

Good luck!