Assignment 1

Course 55807

Due date: **28.01.** The assignment must be submitted electronically by **midnight** on the due date.

Instructions – Please read carefully

1. Assignments should be submitted electronically, as MS Word, PDF document or Jupyter notebook containing text and embedded graphics.
2. Label each plot, chart or table, add detailed title and reference it in your text (i.e. "Figure 1 shows … ").
3. The document can be written in English or Hebrew.
4. Each plot should be complimented with detailed explanation of what it shows, and your conclusions about it.
5. Submitted report, code and data files should be zipped into a single file and uploaded to Moodle. Do not include the data provided to you as part of this assignment.
6. Make sure to provide outputs to your code when feasible (i.e., show the results of your code)
7. **A full answer contains the code you have developed, as well as detailed explanations regarding your <u>own</u> conclusions about the results.**
8. Answer all the verbal questions you were asked.
9. Make sure your code is submitted without errors.
10. Provide explanations (comments) to the functions in your code.
11. The deadline is firm. Late submissions will not be allowed.
12. If you choose to use a model that was not studied in class, provide a short explanation about it.
13. **Guidelines for using ChatGPT (and similar AI generative tools):**
    a. First and foremost, you are responsible for everything you submit.
    b. You are encouraged to use AI with code generation. However, keep in mind that automatically produced code may not suit your needs or could be inefficient.
    c. In cases where you use generative AI model to answer questions, draw conclusions or write explanations, it is essential that you verify its responses yourself and confirm that the text accurately represents your opinion.
    d. It is crucial to tailor your responses to address the particularities of the given data and exercise, rather than providing generic, AI-generated responses.

Good luck

1. (30%) Implement your own TF-IDF vectorizer. Your objective is to create a function ("tfidf_vectorizer") that would receive a corpus (list of strings) and the minimal word frequency (words that would occur fewer than this number of times in the corpus will be ignored). The function should not use sklearn algorithms directly.
   The function will return:
   a. Vocab – a dictionary with a mapping of words to ids.
   b. Document Frequency: a vector of size *len(vocab)* containing the number of documents holding the corresponding word. Element *i* of the vector would contain the number of documents with the word with id *i*.
   c. Word frequency: a vector containing the number of times each word is found in the corpus.
   d. Matrix of shape *len(documents) x len(vocab)* with TF-IDF values for every document in the corpus.

   This function won't be used for very large corpora (don't use sparse matrices). It must first clean the text (remove capitalization and punctuation).

   Test your function with 'tweets.txt'. Provide the outputs of each step of your function applied to the corpus.

2. (25%) Build a small search engine, a function called "search" to search for tweets based on query. The function would receive the data you've generated in 'tfidf_vectorizer', a list of tweets to search through, a search query (string) and the number of matches to return.
   The function will preprocess the query (same pre-processing as the one you used to compute tf-idf), convert the query into a vector and find the requested number of the most similar tweets.
   The function will return: a list of tweets ranked by their similarity to the query and a list of values representing fit of every result.
   To demonstrate that your code works, submit a few examples of its use. Does your search engine work well? Explain.

3. (15%) Compare performance of the function you have created in 1) to sklearn implementation. Which one is faster and by how much? What are the memory requirements for each? (i.e. how much RAM is consumed to store results for each of the approaches?).

4. (30%) The files 'negative_tweets.txt' and 'positive_tweets.txt' contain negative and positive tweets correspondingly. Use WordCloud component (*pip install wordcloud*) to visualize the difference between the two corpora. Make sure to clean the tweets before processing them. Do results make sense? Explain. Use TF-IDF to support and expand your findings.