Text Retrieval & Search Engine (CP423B)

Assignment 2

Maximum Points: 100

Instructions-

- Here are the paraphrased instructions:
- You are required to work on this assignment in the same group as your project.
- Only Python language is permitted for this assignment.
- Plagiarism will be handled according to the institute's policy.
- Your submission should include a recorded video, README.pdf, and code files. It is important to provide clear and concise comments within the code.
- You are allowed to utilize libraries like NLTK for data preprocessing, as well as NumPy, pandas, and matplotlib.
- In the README.pdf, please outline the methodology, preprocessing steps, and any assumptions made.
- Include a description of your outputs and any analysis conducted (if applicable) in the README.pdf.
- The dataset used for Assignment 1 should also be used for this assignment.

Question 1- [30 points] Positional Index

a. [10 points] Execute the subsequent preprocessing tasks on the provided dataset:

i. Transform the text to lowercase.

ii. Perform word tokenization.

iii. Eliminate stopwords from the tokens.

iv. Remove punctuation marks from the tokens.

v. Eliminate empty space tokens.

b. [10 points] Develop the positional index data structure.

c. [10 points] Facilitate the searching of phrase queries, assuming the query length is equal to or less than 5.

Question 2- [50 points] Ranking and Term-Weighting

TF-IDF Matrix [30 points]
The objective is to generate a TF-IDF matrix for every word in the vocabulary and obtain a TF-IDF score for a given query. TF-IDF consists of two components: Term Frequency and Inverse Document Frequency.

- Computing Term Frequency involves calculating the raw count of the word in each document and storing it as a nested dictionary for each document.

- To calculate the document frequency of each word, we need to find the postings list of each word and then determine the number of documents in each posting list for that word.

- The IDF value of each word is computed using the following formula, with smoothing applied:

IDF(word) = log(total number of documents / (document frequency(word) + 1))

- The Term Frequency is calculated using five different variants:

Weighting Scheme	TF Weight
Binary	0,1
Raw count	f(t,d)
Term Frequency	f(t,d)/∑f(t`,d)
Log Normalization	log(1+f(t,d))
Double Normalization	0.5+0.5*(f(t,d)/ max(f(t`,d))

1. First, apply the preprocessing steps mentioned in Q1 to the given dataset:
   - Convert the text to lowercase.
   - Perform word tokenization.
   - Remove stopwords from the tokens.
   - Remove punctuation marks from the tokens.
   - Remove blank space tokens.
2. Construct the matrix with dimensions (number of documents) x (vocabulary size).
3. Populate the TF-IDF values in the matrix for each word in the vocabulary.
4. Create a query vector with a size equal to the vocabulary.
5. Calculate the TF-IDF score for the query using the TF-IDF matrix. Identify the top 5 relevant documents based on the score.
6. Apply all 5 weighting schemes for term frequency calculation and provide the TF-IDF score and results for each scheme individually.

Cosine Similarity [20 points]

To calculate the cosine similarity score for the query using each TF weighting scheme, and report the top 5 relevant documents for each scheme separately, follow these steps:

1. Utilize the query and document vectors obtained from the TF-IDF Matrix in the previous part.
2. Compute the cosine similarity score for the query using each TF weighting scheme.
3. Identify the top 5 relevant documents for each scheme separately based on the cosine similarity score.

Note: For each scoring scheme, provide a report stating the pros and cons of using that particular scheme to determine document relevance.

Hint: The equation for cosine similarity can be represented as follows:

Cosine Similarity = $(A \cdot B) / (||A|| * ||B||)$

Where:
- $A \cdot B$ represents the dot product of vectors A and B.
- $||A||$ represents the Euclidean norm (magnitude) of vector A.
- $||B||$ represents the Euclidean norm (magnitude) of vector B.

Using this equation, the cosine similarity score can be calculated for comparing the query vector and the document vector of length of the vocabulary.

Note: [20 points]

1. Record a video demonstrating your system. Evaluate your system against some phrase queries. Marks will be awarded based on the correctness of the output.
2. Your query output should include:
   - The number of documents retrieved.
   - The list of retrieved document names.
3. Perform preprocessing on the input query as well.

Good luck!