CSE508: Information Retrieval

Assignment 2

Maximum Points: 100

Instructions-

- The assignment is to be attempted in groups (same group as your project)
- Language allowed: Python
- For plagiarism, institute policy will be followed
- You need to submit README.pdf and code files. The code should be well commented.
- You are allowed to use libraries such as NLTK for data preprocessing, NumPy, pandas and matplotlib.
- Mention methodology preprocessing steps, and assumptions you may have in README.pdf.
- Mention your outputs, analysis done (if any) in Analysis.pdf
- You will be required to use Github for code management.
    - Each group will create a GitHub repository with the name IR2021_A1_GroupNo (Eg - IR2021_A1_1 for Group No-1).
    - Each group would add the assigned TA as a collaborator to the GitHub repository. TAs' GitHub handles would be shared shortly.
    - While uploading on Classroom, each group would need to upload a link of the GitHub repository. Only one member needs to submit.
- You will have 10 days to complete the assignment.

Question 1- [15 marks] Positional Index

You are supposed to use the same dataset as used in Assignment 1 for this question.
Download the stories dataset from the given link: http://archives.textfiles.com/stories.zip
The data set consists of 467 files and has a size of about 15MB (including SRE and remaining files). The Farnon folder is excluded from the dataset. Ignore index.html in the stories folder.

a. [2.5 marks] Carry out the following preprocessing steps on the given dataset
    i. Convert the text to lower case
    ii. Perform word tokenization
    iii. Remove stopwords from tokens
    iv. Remove punctuation marks from tokens
    v. Remove blank space tokens
b. [2.5 marks] Implement the positional index data structure
c. Provide support for the searching of phrase queries. You may assume query length to be less than or equal to 5.
d. [10 marks] During the demo, your system would be evaluated against some phrase queries. Marks would be awarded based on the correctness of the output.

Your query output should include:

- The number of documents retrieved
- The list of document names retrieved

Note-

- Perform preprocessing on the input query as well.

Question 2- [60 points] Scoring and Term-Weighting

Jaccard Coefficient [20 points]

The goal is to find the Jaccard coefficient between a given query and the document. The formula used is mentioned below as:

Jaccard Coefficient = Intersection of (doc,query) / Union of (doc,query)

The high the value of the Jaccard coefficient, the more the document is relevant for the query.

1. To calculate this make set of the document token and query token and perform intersection and union between the query and each document.
2. Report the top 5 relevant documents based on the value of the Jaccard coefficient.

TF-IDF Matrix [20 points]

The goal is to generate a TF-IDF matrix for each word in the vocab and obtain a TF-IDF score for a given query. TF-IDF has two parts Term Frequency and Inverse Document Frequency.

- Computing Term Frequency involves calculating the raw count of the word in each document and stored as a nested dictionary for each document.
- To calculate the document frequency of each word, find the postings list of each word and subsequently find the no. of documents in each posting list of each word.
- The IDF value of each word is calculated using the formula as mention below:

  Using smoothing:-
  IDF(word)=log(total no. of documents/document frequency(word)+1)

- The Term Frequency is calculated using 5 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. Use the same data given in Q1 and carry out the same preprocessing steps as mentioned before.
2. Build the matrix of size no. of document x vocab size.
3. Fill the tf_idf values in the matrix of each word of the vocab.
4. Make the query vector of size vocab
5. Compute the TF-IDF score for the query using the TF-IDF matrix. Report the top 5 relevant documents based on the score.
6. Use all 5 weighting schemes for term frequency calculation and report the TF-IDF score and results for each scheme separately

Cosine Similarity [20 points]

The cosine similarity is used to measure the similarity between the query vector and the document vector of length of the vocabulary.

1. Use the query and document vectors obtained from the TF-IDF Matrix in the previous part.
2. Calculate the cosine similarity score for the query using each TF weighting scheme given above and report the top 5 relevant documents for each scheme separately.

Note- State the pros and cons of using each scoring scheme to find the relevance of documents in your report.

Question 3- Ranked-Information Retrieval and Evaluation [25 points]

Use the data file provided here. This has been taken from Microsoft learning to rank dataset, which can be found here. Read about the dataset carefully, and what all it contains.

1. Consider only the queries with qid:4 and the relevance judgement labels as relevance score.

2. [10 points] Make a file rearranging the query-url pairs in order of max DCG. State how many such files could be made.
3. [5 points] Compute nDCG
    a. At 50
    b. For the whole dataset
4. [10 points] Assume a model that simply ranks URLs on the basis of the value of feature 75 (sum of TF-IDF on the whole document) i.e. the higher the value, the more relevant the URL. Assume any non zero relevance judgment value to be relevant. Plot a Precision-Recall curve for query “qid:4”.