Part 2: Indexing and Evaluation

Indexing

1. **Build inverted index:** After having pre-processed the data, you can then create the inverted index.
   HINT - you may use the vocabulary data structure, like the one seen during the Practical Labs:
   {
   Term_id_1: [document_1, document_2, document_4],
   Term_id_2: [document_1, document_3, document_5, document_6],
   etc…
   }

   Documents information: Since we are dealing with conjunctive queries (AND), each of the returned documents should contain all the words in the query.

2. **Propose test queries:** Define five queries that will be used to evaluate your search engine (e.g., “covid pandemic”, “covid vaccine”)

   HINT: How to choose the queries? The selection of the queries is up to you but it’s suggested to select terms based on the popularity (keywords ranked by term frequencies or by TF-IDF, etc...).

3. **Rank your results:** Implement the TF-IDF algorithm and provide ranking based results.

Evaluation

- There will be 2 main evaluation components:
  1. A baseline with 3 queries and the ground truth files for each query will be given to you, using a subset of documents from the dataset.
     a. Query 1: Landfall in South Carolina
     b. Query 2:Help and recovery during the hurricane disaster
     c. Query 3:Floodings in South Carolina

  2. You will be the expert judges, so you will be setting the ground truth for each document and query in a binary way for the test queries that you defined in step 2 at the indexing stage.

- For the prior evaluation components you must evaluate your algorithm by using different evaluation techniques and only for the second component (your queries) comment in each of them how they differ, and which information gives each of them:
  - Precision@K (P@K)
  - Recall@K (R@K)
  - Average Precision@K (P@K)

- F1-Score
- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gain (NDCG)

- Choose one vector representation, TF-IDF or word2vec, and represent the tweets in a two-dimensional scatter plot through the T-SNE (T-distributed Stochastic Neighbor Embedding) algorithm. To do so, you may need first to represent the word as a vector, and then the tweet, i.e., resulted as the average value over the words involved. Any other option rather than T-SNE may be used, but needs to be justified.

  HINT: You don't have to know all the theoretical details used in T-SNE, just use the proper library and generate the output and play with it.

  Also, you can choose to perform an alternative method to generate a 2-dimensional representation for the word embeddings (like PCA).

  Here some T-SNE examples which may be good guidelines for the task:

  1. https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d

  2. https://towardsdatascience.com/visualizing-word-embedding-with-pca-and-t-sne-961a692509f5

  3. https://stackoverflow.com/questions/40581010/how-to-run-tsne-on-word2vec-created-from-gensim