Report Part 2: Indexing and Evaluation.

In this section, we use the following data: a dataset of processed tweets containing information for each tweet, including Tweet, Date, Hashtags, Hashtag_count, Likes, Retweets, URL, and docId. Additionally, an evaluation dataset will be used in the evaluation phase of the project, which includes docId, query_id, and label. We also use the original tweets dataset during the evaluation phase.

We modified the Part 1 .ipynb by saving the processed tweets dataset into a .csv file. This allows us to import it in this section, enabling us to work in separate files for each part of the project. This approach speeds up execution and makes the results clearer to follow in each section.

Indexing.

1. Build an inverted index.

In order to implement the inverted index, we need to build a function that generates the keycomponents for a TF-IDF index: the index, mapping each term to the documents (tweets) it appears in, the term frequency, showing the raw frequency of each term within individual tweets, the df document frequency, indicating how many tweets contain each term and idf which scores terms based on their uniqueness across tweets as a measure of relevance.,

Given the dataset size, over 48,000 tweets, the create_index_tfidf function takes more than three hours to execute on the full dataset. To avoid this time cost with every new session, we serialized the resulting data by saving it in a pickle file, allowing us to simply load the data whenever we run the notebook, this means that we only have to execute the create_index_tfidf one time.

2. Propose test queries.

When evaluating a search engine, it's essential to use a diverse set of search terms that represent various topics and formats to ensure the engine is tested across a range of real-world scenarios users commonly encounter. To achieve this, we selected popular search terms relevant to our context. Below are the five queries we'll use for this evaluation:

- "Indian government response to farmers"
- "International support for farmers"
- "Demands of farmers' protests"
- "Police action during farmers protest"
- "Schedules and locations of demonstrations and protests"

These queries were chosen to test the search engine's performance on different aspects of a single topic, assessing its ability to retrieve comprehensive and accurate information.

3. **Rank your results.**

For each query, we start by processing it, here we use the function defined in first part of the project. Then we use the following functions for rank the results.

1. rank_documents(terms, docs, index, idf, tf): This function ranks documents based on TF-IDF scores, using the index, IDF, and TF values. It calculates the relevance of each document to the query terms.
2. search_tf_idf(query, index): This function searches for the processed query within the inverted index by applying rank_documents, which returns the documents ordered by their relevance to the query.

We test this implementation using the queries defined above and store the resulting ranked documents in a dictionary, making it easy to use this data later for evaluation purposes.

The search results show the top 10 documents retrieved for each query, ranked by their relevance scores. In the Jupyter notebook we can see the top 10 for each one. Here some destacable resulst:

- "Indian government response to farmers": the top result is doc_3234 with the highest score of 11.8045, indicating this document is likely highly relevant to the query, with subsequent documents scoring between 6.7 and 9.5.
- "International support for farmers": doc_859 and doc_21846 both have a high relevance score of 10.9298, suggesting strong alignment with the search terms. Other results have progressively lower scores, down to 7.3, indicating varying degrees of relevance.
- "Demands of farmers protests": doc_15908 stands out with a score of 12.1176, while other documents have lower scores (mostly around 7), suggesting doc_15908 may contain the most specific or comprehensive information about the farmers' demands.
- "Police action during farmers protest": doc_24374 and doc_13980 have the highest score of 11.4668, with relevance scores decreasing after that, suggesting a concentration of relevant content in the top few documents.
- "Schedules and locations of demonstrations and protests": doc_43892 has a notably high score of 22.7138, with other top results scoring between 9.7 and 16, indicating doc_43892 may contain very specific scheduling or location details.

Evaluation.

1. **Evaluation techniques.**

With the defined queries, we first import the evaluation.csv file specified in the project definition. We then separate the evaluation set into distinct subsets for each query. For each subset, we create an index and perform a search to obtain the relevance scores of each document in the respective sets, as we did in the task 1 of the project. Following this, we execute various evaluation techniques to compare the retrieved docIds with those in the ground truth, so, the scores of each document with the label column of the evaluation dataset, allowing us to assess the accuracy and effectiveness of the search engine's results for each query.

For our queries, we act as the expert judges, which, in the context of this practice, means that we need to generate the evaluation file. To accomplish this, we take the ranked documents and results for each query obtained in the ranking part of the practice. This process yields different retrieved documents along with their respective scores for each query. We then select the first 30 retrieved documents for each query and their ranks. Theoretically, all of these documents are relevant to the query, but since we are the expert judges, we will manually verify this by printing the query and the content of the corresponding docIds, i.e., the retrieved tweets. We use the original tweets, without preprocessing, for this assessment because they represent the actual, human-readable content.

We will present an example of a docId that the system retrieved and assess it as relevant for the user, along with another docId that the system retrieved with a high score but which we have determined is not relevant to the query with the main topic of the tweet.

Query 1.

- Relevant: doc_34963 – Why there is no Response of government about the farmers?
- Non relevant: doc_38114 – Criticizes the government in general but does not mention the response to the protests.

Query 2.

- Relevant: doc_859 – Mentions international support for the farmers' protest.
- Non relevant: doc_25064 – Asks if it is an internal issue.

Query 3.

- Relevant: doc_15930 – Farmers demanding MSP.
- Non relevant: doc_15908 – Does not refer to any specific demands.

Query 4.

- Relevant: doc_14464 – Complains about police inaction in response to attacks on farmers.
- Non relevant: doc_32593 – Mentions 'toolkits' without detailing police actions.

Query 5.

- Relevant: doc_45516 – Schedule of Upcoming MahaPanchayats.
- Non relevant: doc_44233 – Discussion about police action and locating a person, not schedules or locations.

Finally, if we determine that a docId is not relevant to the query, we classify that docId in the evaluation set with the label 0, indicating it is not relevant. Conversely, if it is relevant, we assign it the label 1.

In order to evaluate the system we perform the execution of different tecnhiques evaluation.

"Indian government response to farmers".

- Precision@10 = 0.5: half of the top 10 retrieved documents were relevant. While this indicates some relevance, it also suggests that the retrieval system needs improvement
- Recall@10 = 1.0: system successfully retrieved all relevant documents available for this query, indicating comprehensive coverage of the topic.
- Average Precision@10 = 0.4035: while relevant documents were retrieved, their distribution among the ranks may not be optimal.
- F1-Score@10 = 0.6667: it suggests a reasonable effectiveness of the retrieval.
- MAP with k=10 = 0.8762: this high MAP indicates that the average precision across various retrieval points is good.
- MRR with k=10 = 0.2: this low score suggests that relevant documents may not be appearing in the very top ranks
- NDCG with k=10 = 0.3726: it reflects that while some relevant documents are present, their placement in the results is not ideal.

"International support for farmers".

- Precision@10 = 0.3: 3 of the top 10 retrieved documents were relevant, the system included many non-relevant documents in the top results.
- Recall@10 = 1.0: all relevant documents for this query were retrieved.
- Average Precision@10 = 0.8667: relevant documents were ranked relatively high.
- F1-Score@10 = 0.4615: reflects the trade-off between precision and recall, indicating moderate effectiveness in retrieval.

- MAP@10 = 0.6429: good MAP score, suggesting fairly high average precision across retrieval points.
- Mean Reciprocal Rank@10 = 1.0: a relevant document was retrieved in the first rank position.
- NDCG@10 = 0.4441: shows that relevant documents were not ideally distributed across ranks.

"Demands of farmers protests".

- Precision@10 = 0.7: high precision rate here shows that most of the top documents retrieved were relevant.
- Recall@10 = 1.0: indicates that the system successfully retrieved all relevant documents.
- Average Precision@10 = 0.7546: relevant documents were generally well-ranked.
- F1-Score@10 = 0.8235: the balance between precision and recall here indicates strong retrieval effectiveness.
- MAP@10 = 0.3917: moderate average precision across retrieval points.
- MRR @10 = 0.5: indicates relevant documents were not in the top ranks consistently,
- NDCG@10 = 0.65: reflects a good distribution of relevant documents in the top ranks.

"Police action during farmers protest".

- Precision@10, Recall@10, Average Precision@10, F1-Score@10, MAP@10, MRR@10, NDCG@10 = 0: all metrics are zero due to the complete absence of relevant documents in the top 10, suggesting a failure to capture relevant results for this query, and resulting in zero effectiveness.

"Schedules and locations of demonstrations and protests".

- Precision@10 = 0.3: high proportion of non-relevant documents appeared in the top 10.
- Recall@10 = 1.0 the system retrieved all relevant documents, ensuring complete topic coverage.
- Average Precision@10 = 1.0: relevant documents were retrieved in high ranks, showing an ideal ranking distribution.
- F1-Score@10 = 0.4615: moderate precision with high recall, indicating moderate overall retrieval effectiveness.
- MAP@10 = 0.7254: good MAP, showing high average precision across retrieval points.
- MRR@10 = 1.0: the first retrieved document was relevant, strong initial ranking performance.
- NDCG@10 = 0.469: suboptimal distribution of relevant documents across ranks.

2. **Vector representation and T-SNE.**

Finally, for the last part, we chose Word2Vec to obtain the word vector data. Next, we initialized and configured a t-SNE model to reduce the high-dimensional word vector data to two dimensions for visualization, adjusting parameters such as perplexity and iteration limits to ensure optimal convergence. After fitting the model on the word vectors, we obtained the t-SNE results, which provide a transformed 2D representation of the data. The visual results allowed us to observe clusters and patterns among the words, indicating relationships based on their contextual meanings. Due to the large number of words (48K) in the plot, many are difficult to distinguish. For the same reason, to avoid unnecessary re-execution and reduce time complexity, we saved the plot after its initial creation for future analyses.