Miquel Casas Olivella - U161822
Aina Moncho Roig - U172733
Marina Suárez Blázquez - U172958

Part 2: Indexing & Evaluation

This report is about the second part of our IRWA's project. All the procedures and supporting materials from the first section will be omitted.

This time, the major goal is to carry out the full indexing and evaluation process with the ultimate aim of having the greatest ranking function that, given a query, can provide the tweets that are more pertinent to the query and better match its implicit criteria.

INDEXING

Based on the tweets we have been using and cleaning up in Part 1, now we build our index.
The main function for this part is named `create_index_tfidf` which takes as input the data frame created where we stored the doc_id, twee_id, Tweet, Username, Tweet's date, Hashtags, Likes, Retweets & URLs. The function returns:

- **The inverted index** containing terms as keys and the corresponding list of documents these keys appears in as values.
- **The normalized term frequency (TF)** for each term in each document.
- **The document frequency (DF)**, representing the number of documents each term appears in.
- **The inverse document frequency (IDF)**, which represents the IDF of each term.

This will help us for a later use to rank the collection tweets based on popularity (TF-IDF…)

Along the code, we are asked to rank a certain number of tweets that better match the query we are looking for. To achieve this ,we use a function named `rank_documents` which takes as input a list with the terms of the asked query, a list of the documents to rank the asked query, the inverted index of the data structure, the IDF and the TF. The function just prints back the list of ranked documents.

At the same time, this function relies on another one (`search_tf_idf`), the one that actually looks for the queries. We force the function just to look for those queries that contain all the terms that make up the query. With the following picture we can appreciate the documents that contain the query *"florida wind"* once they are ranked (top-10) and when they're not:

```
Top 10 results out of 887 for the searched query florida wind:

page_id = doc_2950      score = 2.540536110556786
page_id = doc_1493      score = 2.3304000557827624
page_id = doc_2945      score = 2.074244273523032
page_id = doc_32        score = 1.9204901084610542
page_id = doc_1170      score = 1.9204901084610542
page_id = doc_153       score = 1.8551086644393724
page_id = doc_198       score = 1.7961935170791754
page_id = doc_2047      score = 1.7577887386602347
page_id = doc_242       score = 1.72351185825636
page_id = doc_206       score = 1.6934512479022468
```

```
Sample of 10 results out of 887 for the searched query -> florida wind:

page_id = doc_2950
page_id = doc_1493
page_id = doc_2945
page_id = doc_32
page_id = doc_1170
page_id = doc_153
page_id = doc_198
page_id = doc_2047
page_id = doc_242
page_id = doc_206
```

The queries we're using for the search function are: `"florida wind"`, `"rain flood"`, `"hurricane disaster"`, `"it's storming"`, `"storm winds"`.

These queries are made up by the most frequently used terms in the collection, this way when it comes to comparing an unusual query we can observe the score's difference between them both.

EVALUATION

Right after indexing our tweets, we need to evaluate our system.
The dataset we are going to work with is the "evaluation_gt.csv", the evaluation methods we have used to evaluate our system are:

- **Precision @K:** given a query, returns the top-k documents based on the system's precision.
- **Recall @K:** given a query, returns the top-k documents based on system's recall.
- **Average precision @K:** given a query, returns the top-k documents based on system's avg. precision.
- **F1-score:** this is a numeric value determined by $2 * \frac{precision*recall}{precision+recall}$. Is the harmonic mean of our system's precision and recall values.
- **Mean average precision (MAP):** combines the system's recall and precision for the ranked retrieval results. As its name suggests, is the mean of the average precision at K across all instances in the dataset.
- **Mean reciprocal rank (MRR):** another measure that calculates the average of the inverse of the ranks at which the first relevant document was retrieved for a set of queries.
- **Normalized discounted cumulative gain (NDCG):** it'll measure the quality of a set of search results.

Along the document we consider k = 10.
With the stated measure we will evaluate the five queries we have proposed in part 1, the queries will take different scores among them.

Getting a little deeper in each of the metrics and the results we have obtained, we see that the resulting values obtained at Precision@K and Average Precision@K are quite similar for all the queries

The following link will redirect you to our GitHub repository, specifically to the corresponding part. In this case part 2: https://github.com/IRWA-2022/PART2