PART2: INDEXING AND EVALUATION

Now, with the processed data is time to create the index so we can have faster searching, to create the index we will start by obtaining the terms using the build\_terms function from the first part of the project. Then, for each term we will store the document and the positions where the term can be found in that document, we will also store a data structure with the most important information from each tweet (tweet, username, date, hashtags, likes, retweets, URL), this way, we can access faster to the most relevant information about a tweet.

After having the index, we have implemented a search method which given a query and an index it pre-processes the query and looks in the index for the tweets where the term appears and returns a list with these documents.

Now, we have to evaluate this search engine, we will do it by testing 5 queries, the queries chosen were:

* Covid pandemic
* International disaster
* Ritmo de vacunacion
* Percentage de hospitalizados
* Mental health

We chose these queries because …

To apply a TF-IDF ranking to the results we created the method create\_index\_tfidf which given the collection of tweets and the total number of tweets returns the index, the term frequency (tf), document frequency (df) and the inverse document frequency (idf) for each term and also tweet\_index containing the most relevant info for each tweet. Then, the ranking will be performed by the function rank\_documents which will compute the frequency of each term in the query an then it will compute the norm for the query tf. We will use theses for the normalized tf-idf calculation. It will also compute for each doc in each index term of the inverted index the document vector. Finally, for each document vector it will compute the dot product with the query vector which will give the cosine similarity of that document (tweet) with the query and it will return the ids of those documents sorted by score.

As we did with the first index, we will create a search\_tf\_idf function which given a query and an index will return a ranked list of documents (tweets) by pre-processing the query and calling the above function. This can be tested by adding by inserting a query in the corresponding cell.

For the evaluation, we applied different techniques which give different information.

First of all, precision\_at\_k will compute the **precision@k** given the ground truth and the predicted scores, this technique gives the proportion of recommended items in the top-k set that are relevant by setting a threshold k, computing the % of relevant documents in the top k and ignoring documents ranked lower than k.

Secondly, avg\_precision\_at\_k will compute average precision@k

Thirdly, map\_at\_k will compute the mean average precision@k given a search results dataset. These gives the average precision across multiple queries/rankings. It considers rank position of each relevant document k1, k2, …, kr, then for each k it computes the P@K and return the average.

Forthly, rr\_at\_k will compute the reciprocal rank which is used for measuring cases where we want to find only one thing, it computes the

**GITHUB URL:** https://github.com/iliasasskali/Project-IR-WA.git

**TAG:** IRWA-2021-final-project-part-2