IR Project
Part 2 Report

Pol Ayala - (u173483)
Toni Carbonell - (u172950)
Nil Distefano - (u172948)

Context

The aim of this report is to explain how we proceeded with the *indexing* and *evaluation* of our IR System.

Indexing

In order to build the index we took code from the labs and mixed it with the pandas that we build from the first part of the project. So, the only difference is that we build our index based on the tweets pandas we constructed in the Part 1 of the project. So we iterated through the pandas to build this index, accessing elements using *row[1]['element.to.access']*.
The functions that we used for our indexing part are:

- build.terms: To pre-process text (same as in Part 1).
- create.index.tf.idf: Builds an inverted index based on tf-idf.
- rank.documents: Performs the ranking based on tf-idf weights. Returns a list of documents with 'document.id' and 'score'.
- search.tf.idf: Gets the union of documents that contain all the query terms and return the ranking of them.
- search.tf.idf.subset: Same as the function before but we add the condition that the documents must also belong to a certain subset.

Evaluation

Once the index was build, we needed to evaluate our system. In order to evaluate a system, it is necessary to have benchmark. This benchmark should be conformed of a collection of documents, a collection of information needs and human relevance assessments. In our case, we would have two sets collections of (information needs, human relevance assessment): one was given to us, the other was made directly by us, using code that appear as 'commented' on the notebook.

Metrics

The metrics that we used to evaluate our system where the following:

- precision@k: system's precision at the top k documents given a query
- AVP@k: system's average precision at the top k documents given a query
- recall@k: system's recall at the top k documents given a query
- f1@10: f1 score (precision*recall/(precision+recall)) using p@k and r@k
- ReciprocalRank@k: system's reciprocal rank (1/n, n being the ranking of the first relevant document among the returned documents) at the top k documents
- MAP@k: mean average precision at the top k documents given multiple queries

Specification: for any "@k" measure, we used a $k = 10$

Benchmark from the given ground.truths

We had been given a set of queries and, alongside with them, a ground.truths pandas table, where for each query and each document, a *ground truth* had been established (0 for non-relevant, 1 for relevant).
The **results** were the following:

1. Query 'Landfall in South Carolina'
    - Precision@10: 1.0
    - AveragePrecision@10: 1.0
    - Recall@10: 1.0
    - F1@10: 0.5
    - ReciprocalRank@10: 1.0
2. Query 'Help and recovery during the hurricanee disaster'
    - Precision@10: 0.9
    - AveragePrecision@10: 0.83
    - Recall@10: 0.9
    - F1@10: 0.45
    - ReciprocalRank@10: 1.0
3. Query 'Floodings in South Carolina'
    - Precision@10: 0.9
    - AveragePrecision@10: 0.89

- Recall@10: 0.9
- F1@10: 0.45
- ReciprocalRank@10: 1.0

4. Multi-level relevance evaluation:
    - MAP@10: 0.9084

Our ground.truths table

We used the same metrics to evaluate our system, this time with our ground.truths table, obtaining the following **results**:

1. Query 'HurracaineIan disaster'
    - Precision@10: 0.8
    - AveragePrecision@10: 0.88
    - Recall@10: 1.0
    - F1@10: 0.44
    - ReciprocalRank@10: 1.0
2. Query 'Damage of HuracaineIan'
    - Precision@10: 0.7
    - AveragePrecision@10: 0.98
    - Recall@10: 1
    - F1@10: 0.41
    - ReciprocalRank@10: 1.0
3. Query 'Florida floodings'
    - Precision@10: 0.9
    - AveragePrecision@10: 1.0
    - Recall@10: 1.0
    - F1@10: 0.47
    - ReciprocalRank@10: 1.0
4. Query 'Storm and wind in Florida'
    - Precision@10: 0.7
    - AveragePrecision@10: 0.83
    - Recall@10: 1.0
    - F1@10: 0.41
    - ReciprocalRank@10: 1.0
5. Multi-level relevance evaluation:
    - MAP@10: 0.8486904761904762
    - NDCG scores: [1.0, 1.0, 1.0, 1.0, 1.0]

Running our code

On the *ReadMe* section of our GitHub *(https://github.com/nildistefano/IRWA-2022-u172948-u172950-u173483)* it can be found how to run this Notebook, in addition to further specifications concerning packages and dependencies.

Results Interpretation & Conclusion

In this lab we programmed some indexing and ranking functions for our IR system, and evaluated them with the above commented metrics. The ranking function that we used was the *tf-idf cos similarity*. Based on the evaluation of our benchmark, both with the given ground.truths and our personal ground.truths, it looks like this ranking function and consequently, our IR system, work pretty well. As commented in class, tf-idf cosine similarity are in the base of IR systems and it has been proved efficient enough for little projects such as ours. In the Part 3 of this project, we will further develop ranking functions in order to strengthen its efficiency.