Part 3: Ranking

Ranking

Ranking score:
Given a query, we want to get the top-20 documents related to the query.

GOAL:
Find all the documents that contain all the words in the query and sort them by their relevance with regard to the query.

SCORE:

1. You're asked to provide 2 different ways of ranking:
    a. **TF-IDF + cosine similarity**: Classical scoring, we have also seen during the practical labs
    b. **Your-Score + cosine similarity:** Here the task is to create a new score, and it's up to you to create a new one.

    Explain how the ranking differs when using TF-IDF and YOUR score and think about the pros and cons of using them.
    Regarding your own score, justify the choice of the score (pros and cons). The only constraint you have is that the score needs to involve the tweets information regarding the popularity over the social network (number of likes, number of tweets, number of comments, etc…)

2. Return a top-20 list of documents for each of the 5 queries, using word2vec + cosine similarity.

    To use the word2vec, you need to generate the tweet representation, which here is expressed as a unique vector of the same dimension of the words, generated as the average of the vectors representing the words included in the tweet:
    Ex. "I won the election"
    Having a vector (generated through word2vec) representing each word, e.g. I=v_1, won = v_2, the = v_3, election = v_4, all of the same number of dimensions n, it is possible to represent the text above and generating a unique representation, by averaging v_1, v_2, v_3, v_4. The result will be a new vector v, of the same dimension n representing the text "I won the election". Since it's a tweet in our case we will talk about tweet2vec.

3. In the context of Information Retrieval, where cosine similarity between vectors is often employed to retrieve relevant documents, how might transformer-based embeddings (such as those from BERT or RoBERTa) enhance or complicate the retrieval process compared to traditional embeddings like word2vec? Consider factors like capturing context within tweets, computational overhead, the potential need for fine-tuning, and the richness of embeddings in representing the nuances of short texts like tweets.