**Part 3: Ranking**

Information Retrieval and Web Analysis

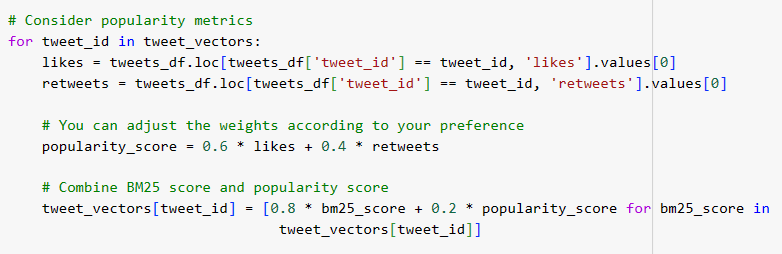
**Introduction**

In this third part of the project, the goal is to find all the documents that contain the words in the query and sort them by their relevance with regard to the query. We will rank these scores in two different ways, using the BM25 ranking algorithm having into account tweets popularity and the Word2Vector model, and then we will compare them to see if one of them is performing better than the other.

**Score**

We are asked to provide two different ways of ranking; one is given: TF-IDF + cosine similarity and one up to us. We decided to use an **extended BestMatch25** (BM25) ranking algorithm, where we take into account the popularity over the social network. This decision was made because BM25 is a simple and effective algorithm that we think combined with popularity information can perform even better than TF-IDF.

In order to create our ranking algorithm we took the base algorithm BM25 and computed the indices as shown in seminars and practicas but when computing the ranking function rank\_documents\_bm25\_with\_popularity we added this extra computations:



This way, the ranking depends not only on the BM25 but also on the popularity score.

The differences we expect between TF-IDF + cosine similarity and our ranking algorithm is that TF-IDF do not take into account the number of likes or retweets, which most of the time leads to ranking tweets higher when the user is saying something non-sense or irrelevant. Considering these likes and retweets we make sure the first positions in the ranking are from relevant people like news media or government officials, which have a higher impact than individual users.

Once both ranking algorithms are implemented, we can compare them. We decided to rank the top 5 for each five of the queries from the previous part, for an easier comparison.

The overall comparison of top retrieved tweets for each query are pretty different. Our ranking algorithm makes the tweets with the best news headlines pop up in the top ranking due to the high interest these have. This is sometimes very useful to see what topic is most supported given a query. It sometimes ranks tweets that have nothing to do with the query, but this happens as well with TF-IDF.

**Top 20 list using word2vec**

After training the model Word2Vec with our tweets we implemented the following functions:

* create\_index\_word2vec: Function to implement the inverted index and compute word vectors. Returns index (dictionary where keys are words and values are lists of tweet\_ids containing that term) and word2vector (dictionary where keys are tweet\_ids and values are the average word vectors for the words in the corresponding tweet, based on the Word2Vec model). Function to implement the inverted index and compute word vectors using Word2Vec
* rank\_documents\_word2vec: To perform the ranking of the results of a search based on word vectors. It calculates a query vector based on the average word vectors of the query terms and it scores each tweet based on the dot product between its word vector and the query vector and then sorts the tweets based on their scores in descending order. Returns this list and the corresponding scores.
* search\_word2vec: To perform the ranking of the results of a search. Is implemented to take a query, find relevant tweets for each query term using an index, combine them, and then rank the combined set of tweets based on their similarity to the query using Word2Vec word vectors. It returns a list of tweets sorted by relevance and their corresponding scores, that is exactly what we need to make this top-20.

**Transformer-based embeddings**

**Do transformer-based embeddings enhance or complicate the retrieval process compared to word2vec?**

Transformer embeddings are great for capturing context and providing fine-grained representations, especially in the context of short texts like tweets. However, computational overhead, token limitations, and potential fine-tuning requirements are a challenge and highly dependent on the specific constraints of the information retrieval task:

On one hand, BERT and RoBERTa retrieve information considering the entire context of a word in a document, capturing different meanings for the same set of words and the semantic similarity. All these features allow us to understand the context in tweets and also a more accurate representation of the relationships between words and phrases, improving the relevance of retrieved documents.

On the other hand, transformers are computationally more intensive compared to Word2Vec due to the amount of parameters and the resources needed. They also have a token limitation that prejudices the retrieval of short texts like tweets. Fine-tuning requires adapting the model to specific characteristics of the information retrieval task on tweets, which introduce additional complexities.

Finally, compared to Word2Vec, transformer embeddings typically have higher dimensions. While this can capture more information, it may also introduce more computational overhead and memory requirements.

**GitHub** [**link**](https://github.com/PatriciaGaray/IRWA-2023-u242781-u192945-u190355)

[**https://github.com/PatriciaGaray/IRWA-2023-u242781-u192945-u190355**](https://github.com/PatriciaGaray/IRWA-2023-u242781-u192945-u190355)

TAG: IRWA Project Part 3