Deliverable Lab 3 CAIM

Arnau Cinca Roca, Enric Rubio Pacho

November 6, 2019

1 Rocchio's rule

1.1 Introduction

The goal of this session is to program a script Rocchio.py which implements a User Relevance Feedback. In order to achive our goal we are going to make use of the Rocchio's rule.
Rocchio feedback approach was developed using the Vector Space Model. The algorithm is based on the assumption that most users have a general conception of which documents should be denoted as relevant or non-relevant.Therefore, the user's search query is revised to include an arbitrary percentage of relevant and non-relevant documents as a means of increasing the search engine's recall, and possibly the precision as well. The number of relevant and non-relevant documents allowed to enter a query is dictated by the weights of the a, b, c variables listed below in the Algorithm section.[1]
However, since we are not going to ask the user which of the queried documents are relevant or not, our script will implement Pseudo-relevance Feedback. So we have to compute the following equation:

$$Query' = \alpha \cdot Query + \beta \cdot \frac{\sum_{i=1}^{k} d_i}{k} \tag{1}$$

Where the second part of the equation is computed by adding the tfidfs vectors from each file obtained from the *Query*.

1.2 Experiments

In order to implemtn the script, we make an *nrounds* loop. In each iteration we perform a query to *elastic search* and with the given files, we compute the tfidfs vector using functions programmed in the previous session. However, to add those vectors we used dicctionaries since mergin vectorst will have a cost of $\mathcal{O}(n \cdot log(m))$ instead of $\mathcal{O}(n)$. Furthermore, to create the *Query'*, we get the R higher weigths from the k most relevant documents and finally we build the equation **(1)**.
Also, we created a dicctionary that stores the *Query* (words with its weigths).

1.3 Observations and Conclusions

Once implemented Rocchio's rule, we runned our script with the newsgroup's collection and with *toronto* as query. The results obtained went from 359 documents on the first round to 7 on the second and, finally, we got the same number of documents in the third round as in the second one, which we

[1]Rocchio's rule definition extracted from *https://en.wikipedia.org/wiki/Rocchio_algorithm*

can clearly see that using Rocchio's rule improves our precision in the searched query.
Furthermore, we executed it with *toronto* and *science* and found that the obtained documents talk about polithics, but there the word *science* only occurs in *Department of Computer Science of University of Toronto.* This is because we are not asking for the user opinion. Moreover, the resulting documents are contain the same text so the most relevant words obtained have three times more relevance.