CAI Lab4

Jose Pérez Cano & Jordi Puig Rabat

October 2020

1 Checking different

It is obvious that if we change the relevance of a term in a query, the order of the result will differ. One other thing we observe that may not be that obvious is that the number of documents in the response don't change if we maintain the different terms in the query although we change the relevance of these terms.

2 Implement Rocchio's law in python

There are many optimizations done in the code. To begin with, there is the difference between using dictionaries instead of ordered vectors. Merging ordered vectors has cost $O(m + n)$ where m and n are the size of each vector. However, the sum of two documents in dictionary form is $O(\min(n, m) \log(\max(n, m)))$. If m or n is little and the other is huge, using dictionaries can reduce the cost of adding two documents.

Another optimization is to prune the list of terms for each documents. But we encounter two possibilities here. If we prune before adding up, we may ignore some important terms. And if we prune after we aren't optimizing the sum. There is another problem associated with the fact that we are using an AND query. Having many non-zero terms will make the query return nothing. The solution to this can be to prune by R before and by r after. Being R big and r little. This way we ensure the sum is closer to the true vale and when doing the query we don't have too much words.

Pruning a dictionary can be optimized too by using a minHeap as we saw in class. In python the implementation needs this header `from heapq import heapify, heappush, heappop` which are the main functions to work with a heap. This way the cost of pruning by R is $O(n+R\log(R)\log(\frac{n}{R}))$ which is better than a naive approach is the size of the dictionary is huge compared to R.

3 Experimenting

After changing the values of the different variables, the first thing we notice is that the value range of SCORE can change a lot. When we increase the number of rounds or the number of new terms the range increases, alternatively when we increase the number of documents to consider relevant, this range decreases. When we change the alpha and beta value, this range doesn't change much. These queries were made over the `20_newspaper` folder. As we don't know what the content is, we can't determine if the results are better or not, but we can assume that a higher SCORE means higher relevance although it may not be lineal on the value.

It goes without saying that each time we get different results in terms of number of documents and ordering. We tried to make a new index with very short, known documents. The results were not what we expected because the relevance ordering is not accurate at all. For instance, we copied some definitions from google to documents and searched for the term `computer`, the most relevant document is computer science definition and not computer definition (what we think should be the most relevant).

With this we conclude that sometimes what an user thinks relevant may be something subjective, but it doesn't mean that it can be modeled with this kind of process although sometimes we get little errors.