UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

CAIM Lab, Session 3:

User relevance feedback

Fall 2021-22

Marco Madalin Farcas Vinter
Gerard Caravaca Ibañez

Index

1.Implmentation of URF

Our implementation follows the schema provided as it simplified the task of thinking how to build the structure of the program. We can divide our process of building the code in three parts, the first was to understand what IndexFilePreprocess.py does and how to modify it; it resulted in no problem as the code was pretty clean and understandable. The second part was using and adapting the TFIDF functions that we programmed last session for the need of this one, that is using dictionaries. And the last one was understanding how Rocchio's algorithm works so we can make the implementation based on dictionaries in order to make it efficient. Our reason when choing the implementation was that the cost of merging two vectors of size n and m is O(n + m) and we keep nhits = k documents from the response; if the size of all the document vectors is n then the cost of merging the remaining documents should be O(kn). Now in opposition, maps are usually implemented with BST, so the cost of search and insert are lower than a list (we refer to python-like lists, not linked lists), that is O(log(n)) for dictionaries vs O(n) for lists. From this we get that merging two dictionaries could cost m∗ log(n) which in theory is higher than O(n+m), but from the previous sessions, we know that most of the terms are repeated (i.e m << n), so in the practice merging dictionaries would be much faster than normal lists.

2.Experiments

In our experiments we varied the different values that affect the URF, tha is α, β, k, R y nrows for different queries with the news index. All the experiments are attached in the folder results, we won't be showing them here as the output obtained is very extensive as a result unbearable to show in a report, nevertheless we are going to discuss the results obtained.

2.1.Varying weights of terms

If we keep R = 2 in the query, that stops terms from being added to it.

3.1 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 1 --B 1 --query toronto nyc

3.2 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 2 --B 1 --query toronto nyc

3.3 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 1 --B 2 --query toronto nyc> ./results/3.3

3.4 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 3 --B 2 --query toronto nyc> ./results/3.4

3.5 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 4 --B 1 --query toronto nyc> ./results/3.5

3.6 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 10 --B 1 --query toronto nyc> ./results/3.6

3.7 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 1 --B 10 --query toronto nyc> ./results/3.7

We see that the higher the alpha, the greater the importance (exponent) that is given to each term in the new query, in opposition if beta is higher less importance is given to the user's terms; another observation is that from round 3 onwards the changes are quite insignificant.

2.2.Varying R and number of documents

4.1 python3 rocchio.py --index 20groups --k 10 --R 1 --nrounds 10 --A 2 --B 1 --query toronto> ./results/4.1

4.2 python3 rocchio.py --index 20groups --k 10 --R 2 --nrounds 10 --A 2 --B 1 --query toronto> ./results/4.2

4.3 python3 rocchio.py --index 20groups --k 10 --R 3 --nrounds 10 --A 2 --B 1 --query toronto> ./results/4.3

4.4 python3 rocchio.py --index 20groups --k 10 --R 4 --nrounds 10 --A 2 --B 1 --query toronto> ./results/4.4

As we change, increasing the R decreases the number of documents obtained, this is because there are more elements in the query, therefore it will be more difficult for all the terms to be in the same document; this gives us the intuition that as R gets bigger precision increases but recall goes down.

2.3.Varying K

5.1 python3 rocchio.py --index 20groups --k 20 --R 1 --nrounds 10 --A 2 --B 1 --query toronto> ./results/5.1

5.2 python3 rocchio.py --index 20groups --k 200 --R 1 --nrounds 10 --A 2 --B 1 --query toronto> ./results/5.2

5.3 python3 rocchio.py --index 20groups --k 500 --R 1 --nrounds 10 --A 2 --B 1 --query toronto> ./results/5.3

5.4 python3 rocchio.py --index 20groups --k 800 --R 1 --nrounds 10 --A 2 --B 1 --query toronto> ./results/5.4

Increments in k gives us more documents therefore recall increases. In this case it is possible that for very high values of k many false positives might be added to the query and therefore we'll get a decrease in precision.

2.4.Varying nrounds

6.1 python3 rocchio.py --index 20groups --k 20 --R 5 --nrounds 10 --A 2 --B 1 --query toronto> ./results/6.1

6.2 python3 rocchio.py --index 20groups --k 20 --R 5 --nrounds 5 --A 2 --B 1 --query toronto> ./results/6.2

6.3 python3 rocchio.py --index 20groups --k 20 --R 5 --nrounds 2 --A 2 --B 1 --query toronto> ./results/6.3

The only noticeable observation we made was that from nrounds = 3 onwards there was little to no change in query result, this might be because the other values take preference.

3.Conclusions

Without repeating ourselves, as some of the conclusion were presented as discussions of each experiment, the main points we observed were:

- Varying K and R contributes to changes in precision and recall
- Nrounds does not affect much the result of queries
- α and β play with the users value in the queries