Lab Session 4: User Relevance Feedback

Alex Carrillo, Laia Albors
Cerca i Anàlisi de la Informació, Ciència i Enginyeria de Dades (UPC)

October 7, 2019

1 Document relevance

As a case of study, we tried the following queries with the `20_newsgroups` corpus. It should be noted that the default `standard` tokenizer and the `lowercase snowball` filter have been used for indexing in all tests.

Query \ Document	Doc1	Doc2	Doc3
`toronto nyc`	4.966255	4.1826124	3.6426592
`toronto^2 nyc`	7.0435276	7.048397	5.1663017
`toronto nyc^2`	7.855238	5.4994392	5.761676

As a result, it is observed that the positions of the returned documents change based on the queries. In the same way, when using the boost operator `^` with weight `2` (instead of default weight `1`), the scores increase and the highest ones shift towards the most relevant documents, i.e. *Doc1,Doc2* and *Doc1* for the second and third query, respectively.

Furthermore, as we were testing and inventing new queries some things were deducted. Typically, for queries of a couple of terms, weights tend to be equally distributed among the shown results. Now, for longer queries we can see a more significant variation. Take, for instance, the following inputs:

	Score of the 1st doc	Score of the 5th doc
`the president of the united states of america`	19.730312	15.795621
`the president^3 of the united^2 states^2 of america`	40.069622	32.432854
`the president^4 of the united^3 states^3 of america^2`	57.63904	47.91883

As the number of words increases and the importance of terms is more aggressive, the scores between the most relevant document and the last shown differ more and more. I.e. from a factor of nearly 4 in the first query up to a factor of nearly 10 in the case of the third query.

2 We will, we will Rocchio you

To merge k ordered vectors (with size n) we could take the approach of merging two at a time. Then the first iteration we would go from k vectors to $\frac{k}{2}$, afterwards to $\frac{k}{4}$, and so on. In the first iteration, every merging would take $2n$ (for merging 2 vectors of size n). In this first iteration there are $\frac{k}{2}$ mergings to be done, so the total cost is $O(n \cdot k)$. The next iteration will take the same, $O(n \cdot k)$ (we merge 2 vectors of size $2n$ and we do $\frac{k}{4}$ merges). In total, there would be $O(\log k)$ iterations. So the total cost would be $O(nk \log k)$.

If we use dictionaries: for every relevant document (k in total) we get their vector of weights (that we suppose of size n). For every term for every vector we find the term in the dictionary and increase its value, which has an average cost of $O(1)$ but in the worst case is $O(n)$. That means that, in average, the cost of using dictionaries to merge k vectors is $O(n \cdot k)$ but in the worst case is $O(k \cdot n^2)$.

3 Experimenting

We tried out the query `brain`, and we could see that the words added to create the new query (`neurotransmitt`, `neuron`) were very related to the original one. That makes us conclude that the queries that produce the pseudorelevance feedback make sense. Since we are adding new words to the query so as to find more relevant documents, the mesure that we are increasing is the precision. Because the more words we add, the less documents we get and more sure we are about their relevance.

Let's see what happens if we change the values of the parameters (fixing all except one).

Using the query `brain` and fixing *nrounds* $= 5$, k=5, $R = 5$:

α	β	N° of documents found	Words of the final query (sorted by weight)
0.2	0.6	6	`visual, depict, brain, humbio, neurotransmitt`
0.4	0.6	6	`visual, depict, brain, humbio, neurotransmitt`
1	0.6	6	`brain, visual, depict, humbio, neurotransmitt`
1	0.3	6	`brain, visual, depict, humbio, neurotransmitt`
1	1.2	6	`brain, visual, depict, humbio, neurotransmitt`

So, the values of α and β influence the words that are to be considered for the new query. We can see from the table above that if $\alpha < \beta$, most of the times the words from the original query are no longer the most important ones. That is what we expected because α is the degree of trust on the original query while β is the importance of the new words from the relevant documents.

Using the query `brain neuron` and fixing *nrounds* $= 5$, k=5, $\alpha = 1$, $\beta = 0.8$:

R	rounds	Docs. found	Words of the final query (sorted by weight)
3	5	6	`brain, neuron, visual`
9	5	3	`brain, neuron, depict, visual, neurotransmitt, data, function, effect, electromicograph`
10	1	16	`brain, neuron`*

*This means that no documents where found with the new query so the algorithm stopped and 16 is the answer of the original query.

These values indicate that if we take into consideration too many words for the new query, no documents will be found that match the query. But if we want to improve the original query, we should choose a value for R at least one integer higher that the number of words of the original query. Otherwise, we would obtain new queries with just some of the original words but not any new one. So the optimal range for R would be $[n + 1, n + 5]$ (approximately), where n is the number of words of the original query.

Using the query `brain` and fixing *nrounds* $= 5$, R=5, $\alpha = 1$, $\beta = 0.8$:

k	N° rounds	N° documents found	Words of the final query (sorted by weight)
10	5	6	`brain, visual, humbio, depict, neurotransmitt`
155	5	2	`brain, o, your, you, visual`

From these results, we can conclude that the higher k is, the less related are the new words with the original query. We can also see that if we consider a lot of documents to create the new query, the added words are common ones. This is, words that could be easily found in any document.

Using the query `brain neuron` and fixing $k = 5$, R=5, $\alpha = 1$, $\beta = 0.8$:

nrounds	N° of documents found	Words of the final query (sorted by weight)
3	6	`brain, neuron, visual, depict, neurotransmitt`
10	6	`brain, visual, neuron, depict, humbio`
50	6	`visual, brain, depict, humbio, neurotransmitt`
100	6	`visual, humbio, depict, brain, neurotransmitt`

From the table above, we see that if the number of rounds is high, the words of the original query lose importance in the new one. So, if we are interested in maintaining the original user's query and we just want to complete it with new relevant words, we should not make more than 10 rounds.