Basic IR System

Ignacio Amaya de la Peña

Introduction

Based on the sample IR system provided in class, a more complete IR system have been developed. There are implemented four different combinations of term frequency (TF) and inverse document frequency (IDF) variants. All the logarithms used in the formulas are neperian. Depending on which combination of TF-IDF the user chooses the documents retrieved are in a different order for the same query. That is because of the different weights that are assigned in each method to the words in the documents.

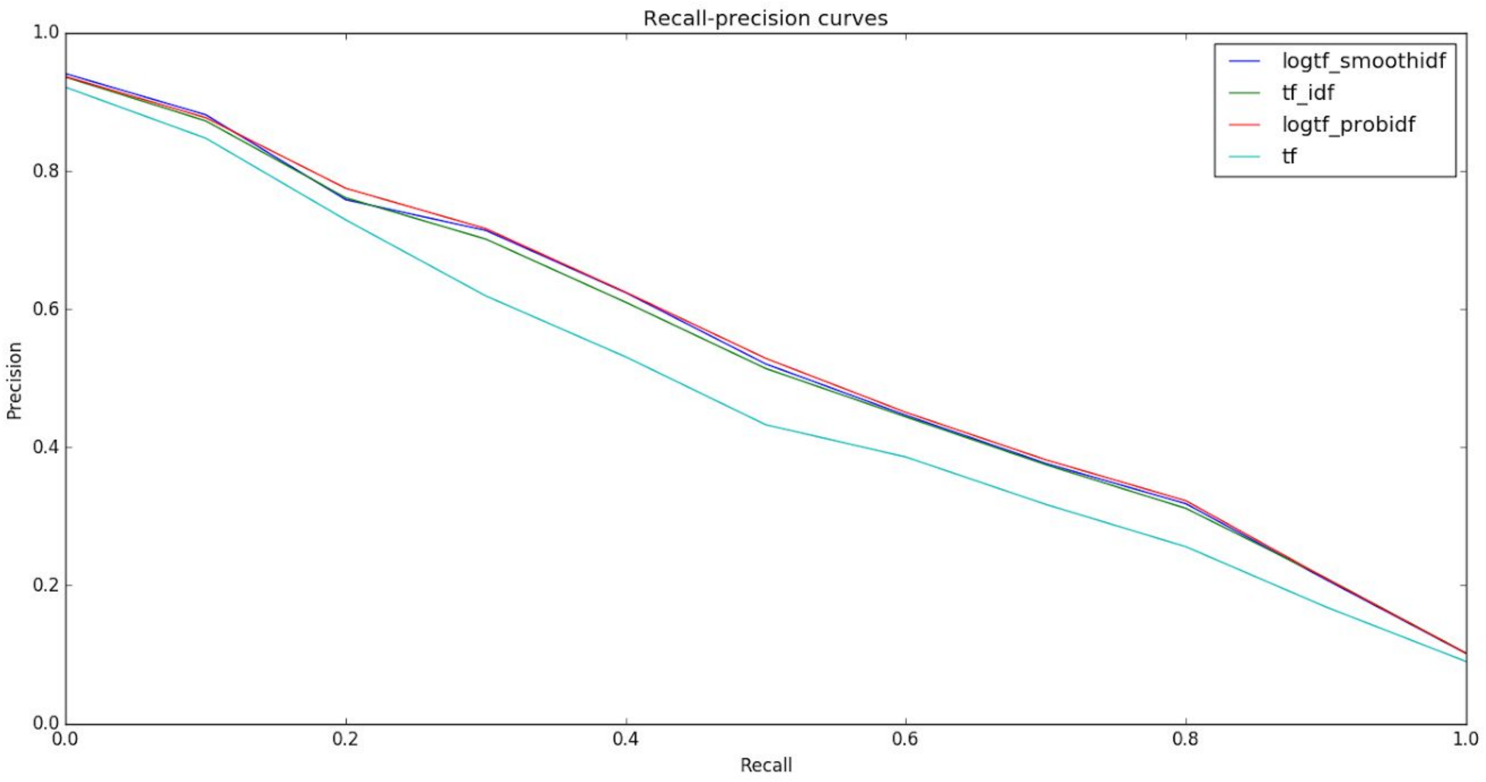
The different combinations are:

- Simple TF without IDF
- TF with log IDF
- Log TF with probabilistic IDF
- Log TF with inverse frequency smooth

There are two python files that can be run using python from the command line: the Metrics.py, which plots the different precision/recall curves to compare the different methods implemented and the Query_launcher.py which allows the user to interact with the IR system. The queries can be inserted by the user or selected from the queries file. The number of the documents retrieved and the TF-IDF model used can also be selected. Once the results are retrieved it is possible to tell the IR system to automatically improve its precision/recall metrics. This is a variant of the Rocchio's method to automatically improve the query results where no input from the user is needed. However, it could be changed to allow the to select the texts he thinks that are relevant. We've preferred to take those texts from the file with the query automatically because of the lack of knowledge in the medical domain, which limited us when selecting the documents. Also, it is important to highlight that in the ranking retrieved only the ids of the documents are shown for clarity, but documents could also be shown if desirable.

Evaluation

Each of the queries provided has been compared with the expected results for each of the four different TF-IDF specified above. The values have been represented interpolating the precision values to a set of 11 recall levels (0 to 1 in intervals of 0.1). In the following plot we can see that the worst results are gotten used plain TF, which was what we expected because of the simplicity of that model. The best approach is to use the log of the tf and the formula for the probabilistic IDF. It is really close to the smoothed IDF one, but it has a greater precision in some points, especially in the 0.2 recall level, which means that some more relevant documents are shown up within the top 20% documents retrieved.



Recall Levels	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Probabilistic	0.94	0.88	0.77	0.72	0.63	0.54	0.45	0.38	0.32	0.21	0.11
Smooth	0.95	0.88	0.76	0.71	0.62	0.53	0.45	0.37	0.32	0.21	0.11
Log TF-IDF	0.94	0.87	0.76	0.70	0.61	0.52	0.44	0.38	0.31	0.21	0.11
TF	0.92	0.85	0.73	0.61	0.53	0.44	0.39	0.32	0.26	0.17	0.10

Automatic query improvement

In order to improve the results obtained by the queries some automatic adjustments can be make to the queries. In this example, as we had the evaluation files, the feedback from the user has been taken directly from files and then we have computed the precision/recall curves to check the improvement made. However, most of the times we don’t have any way to evaluate the corpus, so it will be the user the one saying which documents retrieved are useful and which ones aren’t. Here we’ve considered that all the ones that influentiate in a query are useful and the others are not. This way we assign a negative weight of 0.5 to all the non relevant documents and a weight of 1 to the relevant ones.

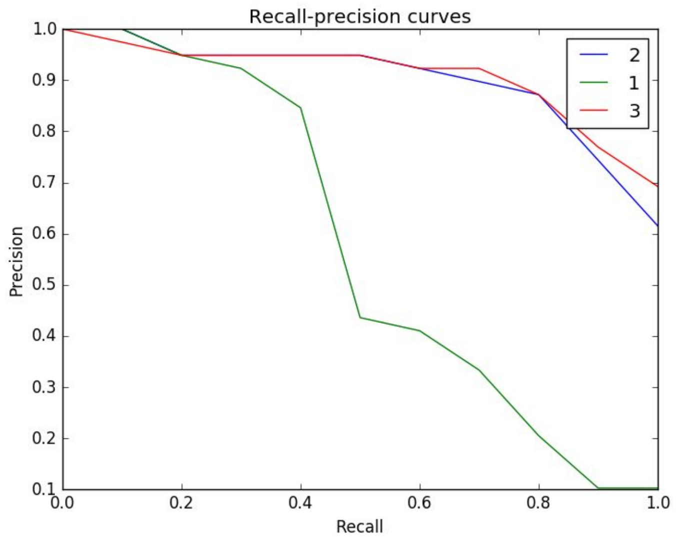
The new query is computed from the weights of each document multiplied by 1 or -0.5 depending on its relevance. This way we are diminishing the importance of non relevant terms in our query, but we also are including other related terms from those documents that can be useful, although i might have not occurred to us to put them in the query.

The user says the number of document that wants to see in the screen and from those the relevant ones are chosen.

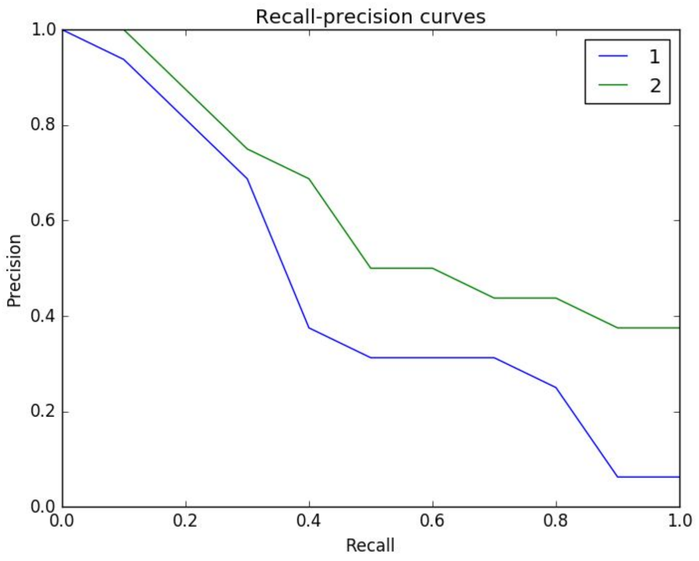
We are going to show several examples of this automatic evaluation showing how the performance of our retrieval system improves. The algorithm implemented stops when when the average precision of the new precision/recall curve is less or equal as the previous one.

We will show now three examples with different values and TF-IDF models.

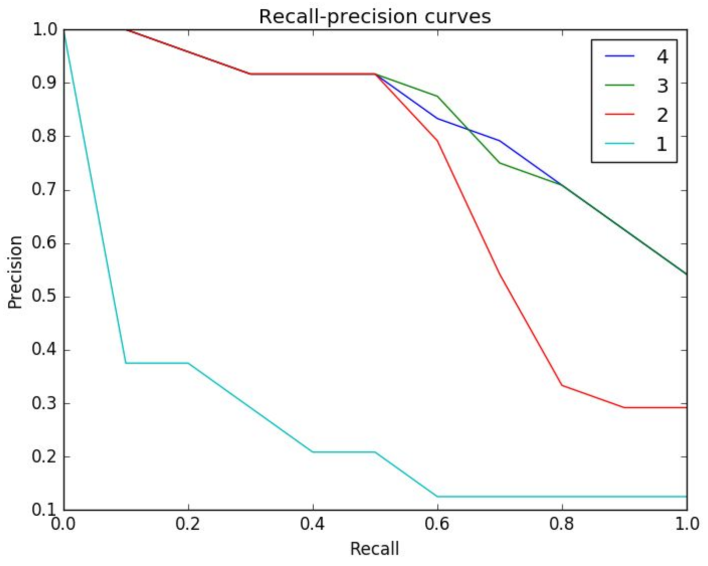
In the first one we have asked for the relevant documents for the query 28 using the probabilistic IDF implementation. From those retrieved we only picked the first 28 to be shown and from those the automatic improvements have been made. We can clearly see that in the first iteration it improves a lot the precision. From the second iteration to the third the change it is not so big, but still noticeable.



In the following plot we've used the smooth method with the second query and only the first two documents in the ranking. As the number of documents is very small, the improvement is not so big as in the previous one.



Finally, we've plotted a TF model for query 10 with the first 30 documents. There are four iterations and the last two ones are really close. We see a big improvement in the precision, especially compared with the first iteration.


Recall-precision curves
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0
4
3
2
1