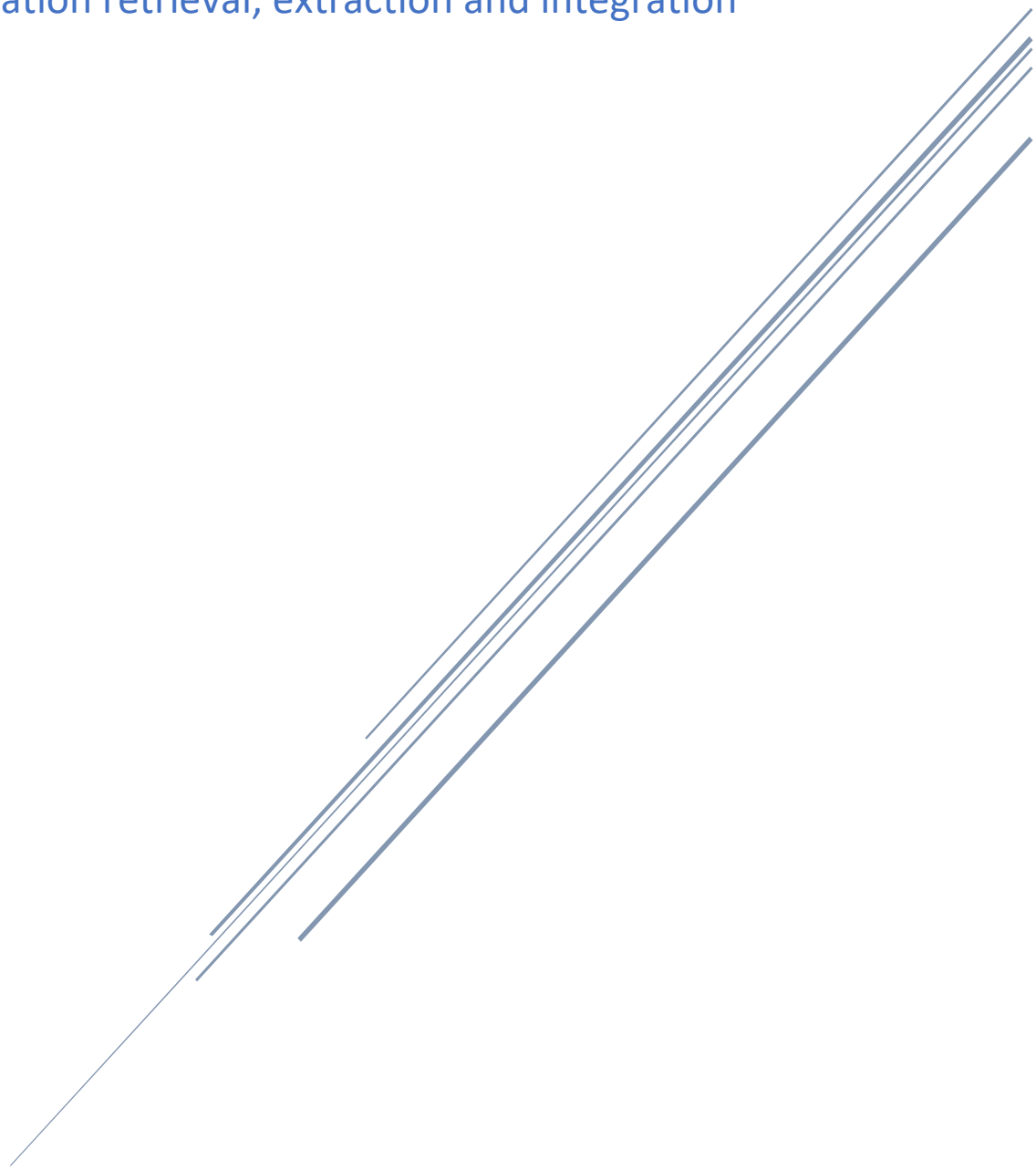
ML RANKING ASSIGNMENT

Information retrieval, extraction and integration



Catalán Gris, Lucía
Garcia De Viedma Pérez, Lucas

Table of contents

1. Introduction

The goal of this project was to build a ranking algorithm that allowed us to determine the relative importance of documents in a dataset with respect to three different medical queries: “glucose in blood”, “bilirubin in plasma” and “white cells count in blood”. Out of the three possible ranking methods given, we decided to use the listwise ranking algorithm "ADARANK".

2. Description of the original dataset

The original dataset included only 67 documents, but since we had to analyze the relationship between these documents and the three queries, it could be considered to have 201 documents instead. This is still a small number of documents for the task, which is why we will increase the size of the dataset in future steps.

Each of the documents includes the following information:

- **Loinc_num**: LOINC ID that differentiates the document from others
- **Long_common_name**: generic name of the document / article
- **Component**: component being analyzed
- **System**: system or medium where the component is being analyzed
- **Property**: type of measurement / property being measured

	loinc_num	long_common_name	component	system	property
0	1988-5	C reactive protein [Mass/volume] in Serum or P...	C reactive protein	Ser/Plas	MCnc
1	1959-6	Bicarbonate [Moles/volume] in Blood	Bicarbonate	Bld	SCnc
2	10331-7	Rh [Type] in Blood	Rh	Bld	Type
3	18998-5	Trimethoprim+Sulfamethoxazole [Susceptibility]	Trimethoprim+Sulfamethoxazole	Isolate	Susc
4	1975-2	Bilirubin.total [Mass/volume] in Serum or Plasma	Bilirubin	Ser/Plas	MCnc

3. Manual ranking process

To rank the documents for each query we determined five possible values: 4 – Perfect result, 3 – Highly related, 2 – Somehow related, 1 – Poorly related, 0 – Nothing in common. The table below sums up the main factors that were considered when assigning these values. The documents with the same value have the same relevance and hence the order between them is irrelevant.

Similarity between query and long_common_name	Query component matches	Query system matches	Value
Yes	Yes	Yes	4
Yes	Yes	No	3
Yes	No	Yes	2
No	No	Somehow similar (f.e Blood & BPO)	1
No	No	No	0

Once this ranking was done for every combination of query-document, we concatenated all of them the following structure (since it’s only the 5 first rows of the dataset there are only query-

document pairs for the first query). *The queries have been encoded as 1, 2 and 3, the columns of the documents as f0, f1, f2 and f3, and the ranking as Y.*

	Query id	f0	f1	f2	f3	Y
0	1	Indirect antiglobulin test complement specific...	Indirect antiglobulin test complement specific...	Ser / Plas	ACnc	0
1	1	Rh Type in Blood	Rh	Bld	Type	2
2	1	Major crossmatch interpretation	Major crossmatch	Ser / Plas	Imp	0
3	1	Methicillin resistant Staphylococcus aureus Pr...	Staphylococcus aureus methicillin resistant is...	XXX	ACnc	0
4	1	Bilirubin total Mass / volume in Synovial fluid	Bilirubin	Synv fld	MCnc	0

4. Obtaining the features for the "ADARANK" model

Since "ADARANK" works with numerical data, we needed to get numerical features that allow the model to predict the relevance of the different documents. To do so we chose to represent each of the original features with their cosine distance to the query after performing TF-IDF over them. This way, the features not only show their importance with respect to the assigned value, but also keep information about the query that is being used, since the TF-IDF and cosine distance values are dependent on it.

However, previous to this step, we need to preprocess the features to improve the metrics obtained by TF-IDF. The preprocessing will consist on setting all the letters in lowercase, changing the found abbreviatures into complete words (f.e. ser -> serum) and removing the stop words.

The state of the dataset after performing the preprocessing and encoding of the features can be seen in the next images.

	Query id	f0	f1	f2	f3	Y
0	1	indirect antiglobulin test complement specific...	indirect antiglobulin test complement specific...	serum / plasma	acnc	0
1	1	rh type blood	rh	blood	type	2
2	1	major crossmatch interpretation	major crossmatch	serum / plasma	imp	0
3	1	methicillin resistant staphylococcus aureus pr...	staphylococcus aureus methicillin resistant is...	xxx	acnc	0
4	1	bilirubin total mass / volume synovial fluid	bilirubin	synovial fluid	mcnc	0

Pre-processing result

	Query id	f0	f1	f2	f3	Y
0	1	0.0	0.0	0.0	0	0
1	1	0.57069	0.0	1.0	0	2
2	1	0.0	0.0	0.0	0	0
3	1	0.0	0.0	0.0	0	0
4	1	0.0	0.0	0.0	0	0

Encoding result

5. Using the "ADARANK" model

The "ADARANK" implementation we've used can be obtained from the following link. It has no documentation, but it follows step by step all the process indicated in the "ADARANK" article we read in class, which can be seen analyzing the code.

To prepare the data for the model, we divide the dataset in 75% training and 25% testing, making sure that all the queries are as equally represented in both of them as possible. In addition, we will separate the features (X) from the ranking labels (Y).

Once the data is prepared, we will train the "ADARANK" model, using the training features (X), ranking labels (Y) and query ids.

```
model = AdaRank(verbose=True)
model.fit(x_sparse,y,qid)
```
✓ 0.2s

```
1       1.3525061929860454      0       [0.97868587 1.          0.64524473]     train 0.8746    valid 0.8746
2       1.2452898801181327      0       [0.97868587 1.          0.64524473]     train 0.8746    valid 0.8746
```

With the trained model we use ONLY the features from the testing dataset and obtain the following predictions.

```
y_te
```
✓ 0.5s

```
array([2, 2, 1, 0, 0, 2, 2, 0, 0, 3, 2, 2, 2, 0, 4, 3, 2, 0, 2, 2, 2, 2,
       1, 0, 3, 0, 4, 1, 1, 1, 1, 1, 1])
```

True values (manual ranking of the documents used for testing)

```
pred
```
✓ 0.1s

```
array([0.77186141, 0.77186141, 0.77186141, 0.        , 0.        ,
       0.77186141, 0.77186141, 0.        , 0.        , 1.11063178,
       0.77186141, 0.63869903, 0.63869903, 0.        , 1.35250619,
       0.        , 0.63869903, 0.        , 0.63869903, 0.63869903,
       0.63869903, 0.63869903, 0.        , 1.35250619, 1.35250619,
       0.        , 1.35250619, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        ])
```

Predicted values (obtained with AdaRank)

Let’s observe the ranking accuracy scoring obtained from the evaluator used by the "ADARANK" model (NDGC), since it’s the one recommended by the article.

```
model.evaluate(y_te, pred, qid_t)
```
✓ 0.1s

```
array([0.98633523, 0.94857094, 0.63258797])
```

This metric focuses on the relative importance the model gave to each document, rather than the similarity between the real values and the predicted values, which is why the obtained results are really good, with the exception of the predictions related to the third query. We can observe how, even though the predicted values don't match at all with the real ranking values, there is a relation between most of the real and predicted values, as documents that had a higher manual ranking value also have a higher predicted importance.

To understand both the results and the resulting metrics we have to consider that most of the documents that are included in the datasets have been tagged as 0 (since only a few documents were really related to the queries) and this creates an unbalance in the predictions. This unbalance causes the predictions to have lower values, which also means that the predictions

will be closer to zero, deriving in better predictions in documents with less relevance (most of them). In addition, the TF-IDF scoring for the 3rd query is not too accurate because in the ranking we also took into account information that we knew and not only similarity (types of white cells, for example).

We tried to obtain a more realistic result by increasing the size of the dataset. We added 195 documents from LOINC to the dataset, prioritizing documents that would get a high value (2,3,4) according to the manual ranking method we previously explained. Our main goal was both to enlarge the dataset to optimize the predictions and to balance a bit more the data to avoid the overload of low values the original dataset had.

6. Comparing the results: original vs enlarged dataset

To compare the results, we will use three metrics. NDGC, to analyze if the relative importance of the documents is captured better by the model, and MSE/MAE to determine the numerical difference between the real and predicted values for each dataset.

6.1. NDGC

Results of the original dataset:

```
model.evaluate(y_te, pred, qid_t)
✓ 0.4s
array([0.99263436, 0.95320391, 0.68004731])
```

Results of the enlarged dataset:

```
model.evaluate(y_te, pred, qid_t)
✓ 0.6s
array([0.91393953, 0.97424523, 0.74748527])
```

We can observe how the increase in size of the dataset has clearly improved the results of predictions regarding the third query, while having small negative variances in the other two.

As we said previously, the results that were previously displayed were surprisingly good, but it was due to the unbalanced number of zeroes in the dataset. Thus, by adding more data we compensate this deviance but also lose some precision predicting lower values, which caused the slight decrease in the two first queries.

We also have to take into account that we are still working with a small dataset, even after the increase, and that a sufficiently bigger dataset (with perfect balance between different ranking values) would show more realistic, and probably better results.

6.2. MAE

Results of the original dataset:

```
The MAE for queries with rank 0 is:
0.171878124279385
The MAE for queries with rank 1 is:
0.8528663757242523
The MAE for queries with rank 2 is:
1.2744785748029819
The MAE for queries with rank 3 is:
2.355457033952306
The MAE for queries with rank 4 is:
2.7109140679046124
The overall MAE is:
1.4731188353327076
```

Results of the increased dataset:

```
The MAE for queries with rank 0 is:
0.09786687395604146
The MAE for queries with rank 1 is:
0.8452306818074918
The MAE for queries with rank 2 is:
1.2058441253416212
The MAE for queries with rank 3 is:
1.739873927964559
The MAE for queries with rank 4 is:
2.4779879794789994
The overall MAE is:
1.2733607177097426
```

The goal of the analysis of this metric is to observe and demonstrate how increasing the dataset has improved the improved the ranking in terms of numerical difference between predicted and real values.

In this case we can see how increasing the dataset has also improved the similarity between the predicted values (in average) and the real ones in every single class, achieving a lower overall too. However, we can see that the MAE for higher ranked queries is still much higher than for the lower ranked ones.

6.3. MSE

Results of the original dataset:

```
The MSE for queries with rank 0 is:
0.2215656720434979
The MSE for queries with rank 1 is:
0.8139742684110891
The MSE for queries with rank 2 is:
1.672544246835402
The MSE for queries with rank 3 is:
5.963613473876954
The MSE for queries with rank 4 is:
7.349055083563133
The overall MSE is:
3.204150548946015
```

Results of the enlarged dataset:

```
The MSE for queries with rank 0 is:
0.11295087607118151
The MSE for queries with rank 1 is:
0.8240934834693735
The MSE for queries with rank 2 is:
1.5038482686432966
The MSE for queries with rank 3 is:
3.3832779342848602
The MSE for queries with rank 4 is:
6.2556090590519995
The overall MSE is:
2.415955924304142
```

Similar to the previous metric, in this case we also want to analyze whether the new dataset has improved the ranking in terms of numerical difference between predicted and real values. MSE increases the relevance of the numerical difference because it works with square values, so it'll allow us to see the improvements from a different angle than the MAE.

Thanks to that we can see more clearly how the increase of the dataset has had a relevant impact on higher ranked queries, improving a lot the results for values 3 and 4, and having a decreasing improvement over values 2, 1 and 0.

5. Checking the rankings: original dataset vs enlarged dataset

In this section we will finish the report by showing an example of the ranking performed by the model with each of the datasets. It can be seen that the extended dataset achieves better rankings, assigning better relative values to the documents and creating more homogeneous rankings overall. *(The following images are ordered according to the Y prediction value)*

Original dataset ranking prediction for the test query-document pairs (query 1 & 2; query 3 in the next page):

	Query id	f0	f1	f2	f3	Y_real	Y_pred
59	1	1.00000	0.671742	1.0	0	4	1.597533
48	1	0.57069	0.000000	1.0	0	2	0.911696
57	1	0.00000	0.000000	0.0	0	0	0.000000
21	1	0.00000	0.000000	0.0	0	0	0.000000
38	1	0.00000	0.000000	0.0	0	0	0.000000
20	1	0.00000	0.000000	0.0	0	0	0.000000
27	1	0.00000	0.000000	0.0	0	0	0.000000
43	1	0.00000	0.000000	0.0	0	0	0.000000
34	1	0.00000	0.000000	0.0	0	0	0.000000
60	1	0.00000	0.000000	0.0	0	0	0.000000
0	1	0.00000	0.000000	0.0	0	0	0.000000

	Query id	f0	f1	f2	f3	Y_real	Y_pred
107	2	0.472234	0.0	1.0	0	2	0.754409
110	2	0.472234	0.0	1.0	0	2	0.754409
126	2	0.472234	0.0	1.0	0	2	0.754409
75	2	0.472234	0.0	1.0	0	2	0.754409
116	2	0.472234	0.0	1.0	0	2	0.754409
99	2	0.472234	0.0	1.0	0	2	0.754409
124	2	0.000000	0.0	0.0	0	3	0.000000
109	2	0.000000	0.0	0.0	0	1	0.000000
76	2	0.000000	0.0	0.0	0	0	0.000000
111	2	0.000000	0.0	0.0	0	0	0.000000
72	2	0.000000	0.0	0.0	0	0	0.000000

	Query id	f0	f1	f2	f3	Y_real	Y_pred
159	3	1.0	0.0	1.0	0	2	1.597533
181	3	1.0	0.0	1.0	0	4	1.597533
169	3	1.0	0.0	1.0	0	2	1.597533
190	3	0.0	0.0	0.0	0	1	0.000000
191	3	0.0	0.0	0.0	0	0	0.000000
175	3	0.0	0.0	0.0	0	1	0.000000
137	3	0.0	0.0	0.0	0	0	0.000000
183	3	0.0	0.0	0.0	0	1	0.000000
134	3	0.0	0.0	0.0	0	3	0.000000
184	3	0.0	0.0	0.0	0	1	0.000000
145	3	0.0	0.0	0.0	0	1	0.000000

Enlarged dataset ranking prediction for the test query-document pairs (query 1, 2 & 3). Only the 20 first results of the ranking are displayed:

	Query id	f0	f1	f2	f3	Y_real	Y_pred
163	1	1.000000	0.46619	0.000000	0	3	3.086688
135	1	1.000000	0.46619	0.268347	0	4	3.086688
104	1	1.000000	0.46619	0.000000	0	4	3.086688
108	1	1.000000	0.46619	0.000000	0	4	3.086688
137	1	1.000000	0.46619	0.268347	0	4	3.086688
120	1	0.852551	0.46619	0.268347	0	4	2.631559
162	1	0.852551	0.46619	0.000000	0	3	2.631559
8	1	0.852551	0.46619	0.000000	0	3	2.631559
231	1	0.522644	0.00000	0.268347	0	2	1.613239
149	1	0.522644	0.00000	0.000000	0	2	1.613239
228	1	0.522644	0.00000	0.268347	0	2	1.613239
52	1	0.522644	0.00000	0.268347	0	2	1.613239
214	1	0.522644	0.00000	0.268347	0	2	1.613239
194	1	0.522644	0.00000	0.268347	0	2	1.613239
84	1	0.522644	0.00000	0.268347	0	0	1.613239
255	1	0.522644	0.00000	0.268347	0	2	1.613239
261	1	0.522644	0.00000	0.268347	0	2	1.613239
37	1	0.522644	0.00000	0.268347	0	2	1.613239
256	1	0.522644	0.00000	0.268347	0	2	1.613239
199	1	0.522644	0.00000	0.268347	0	2	1.613239

	Query id	f0	f1	f2	f3	Y_real	Y_pred
286	2	0.523283	0.000000	1.0	0	2	1.615212
341	2	0.523283	0.865285	0.0	0	3	1.615212
523	2	0.523283	0.000000	1.0	0	2	1.615212
496	2	0.523283	0.000000	1.0	0	2	1.615212
429	2	0.523283	0.000000	1.0	0	2	1.615212
484	2	0.523283	0.000000	1.0	0	2	1.615212
301	2	0.523283	0.000000	1.0	0	2	1.615212
521	2	0.523283	0.000000	1.0	0	2	1.615212
285	2	0.523283	0.000000	1.0	0	2	1.615212
509	2	0.523283	0.000000	1.0	0	2	1.615212
275	2	0.523283	0.000000	1.0	0	2	1.615212
428	2	0.523283	0.000000	1.0	0	2	1.615212
506	2	0.523283	0.000000	1.0	0	2	1.615212
517	2	0.523283	0.000000	1.0	0	2	1.615212
291	2	0.523283	0.000000	1.0	0	2	1.615212
374	2	0.000000	0.000000	0.0	0	0	0.000000
314	2	0.000000	0.000000	0.0	0	0	0.000000
382	2	0.000000	0.000000	0.0	0	0	0.000000
426	2	0.000000	0.000000	0.0	0	1	0.000000
386	2	0.000000	0.000000	0.0	0	1	0.000000

	Query id	f0	f1	f2	f3	Y_real	Y_pred
602	3	1.000000	0.483528	1.0	0	4	3.086688
603	3	0.946908	0.483528	0.0	0	3	2.922811
716	3	0.699720	0.000000	1.0	0	4	2.159816
715	3	0.699720	0.000000	1.0	0	4	2.159816
585	3	0.321503	0.000000	1.0	0	2	0.992380
659	3	0.321503	0.000000	1.0	0	2	0.992380
611	3	0.321503	0.000000	0.0	0	2	0.992380
618	3	0.321503	0.000000	0.0	0	2	0.992380
661	3	0.321503	0.000000	1.0	0	2	0.992380
627	3	0.321503	0.000000	0.0	0	2	0.992380
658	3	0.321503	0.000000	0.0	0	2	0.992380
624	3	0.321503	0.000000	1.0	0	2	0.992380
549	3	0.321503	0.000000	1.0	0	2	0.992380
609	3	0.321503	0.000000	0.0	0	2	0.992380
681	3	0.321503	0.531495	0.0	0	0	0.992380
773	3	0.321503	0.000000	1.0	0	2	0.992380
643	3	0.321503	0.000000	1.0	0	2	0.992380
622	3	0.321503	0.000000	1.0	0	2	0.992380
598	3	0.321503	0.000000	1.0	0	2	0.992380
623	3	0.321503	0.000000	1.0	0	2	0.992380