Catalán Gris, Lucía

Garcia De Viedma Pérez, Lucas

ML Ranking Assignment

Information retrieval, extraction and integration

Table of contents

[1. Introduction 1](#_Toc100955822)

[2. Description of the original dataset 1](#_Toc100955823)

[3. Manual ranking process 1](#_Toc100955824)

[4. Obtaining the features for the "ADARANK" model 2](#_Toc100955825)

[5. Using the "ADARANK" model 2](#_Toc100955826)

[6. Comparing the results: original vs enlarged dataset 4](#_Toc100955827)

[6.1. NDGC 4](#_Toc100955828)

[6.2. MAE 4](#_Toc100955829)

[6.3. MSE 5](#_Toc100955830)

[5. Checking the rankings: original dataset vs enlarged dataset 5](#_Toc100955831)

# 1. Introduction

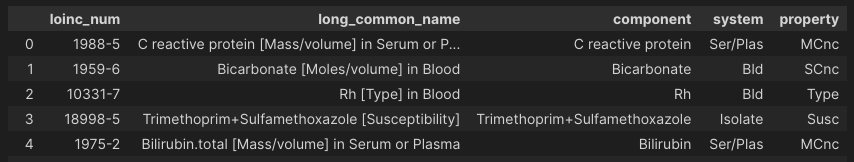
The goal of this project was to build a ranking algorithm that allowed us to determine the relative importance of documents in a dataset with respect to three different medical queries: “glucose in blood”, “bilirubin in plasma” and “white cells count in blood”. Out of the three possible ranking methods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

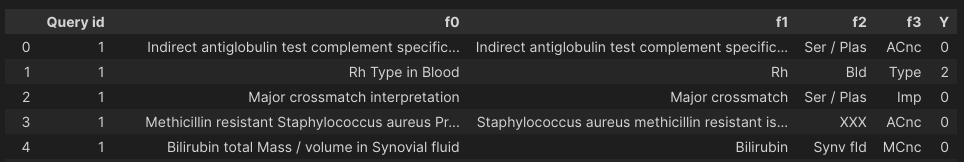


# 3. Manual ranking process

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.*

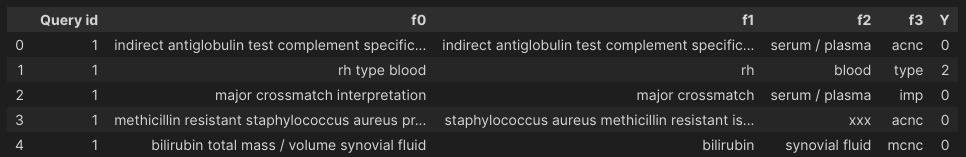


# 4. Obtaining the features for the "ADARANK" model

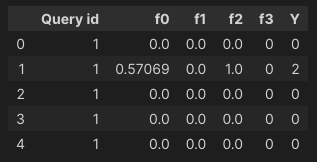
Since "ADARANK" works with numerical data, we needed to get numerical features that allow the model to predict the relevance of the different documents. To do so we chose to represent each of the original features with their cosine distance to the query after performing TF-IDF over them. This way, the 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.



Pre-processing result



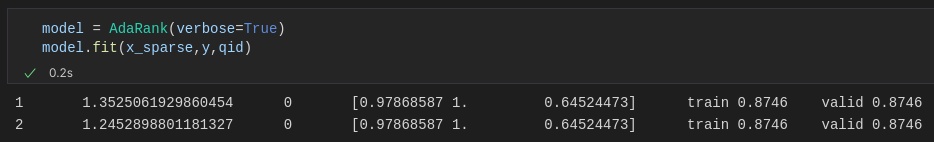
Encoding result

# 5. Using the "ADARANK" model

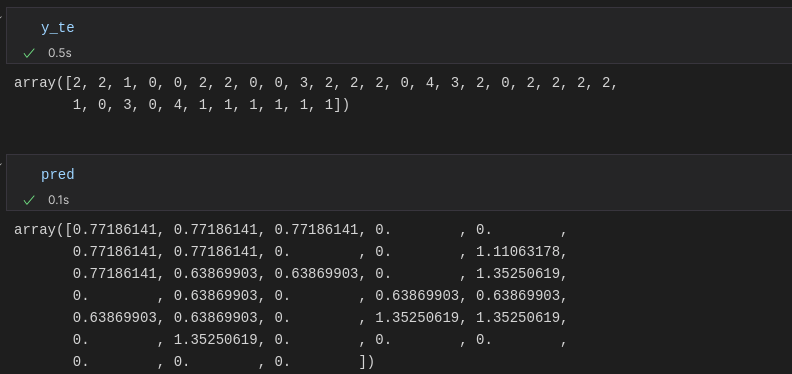
The "ADARANK" implementation we’ve used can be obtained from the following [link.](https://github.com/rueycheng/AdaRank/blob/master/adarank.py) It has no documentation, but it follows step by step all the process indicated in the "ADARANK" article we read in class, which can be seen analyzing the code.

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).

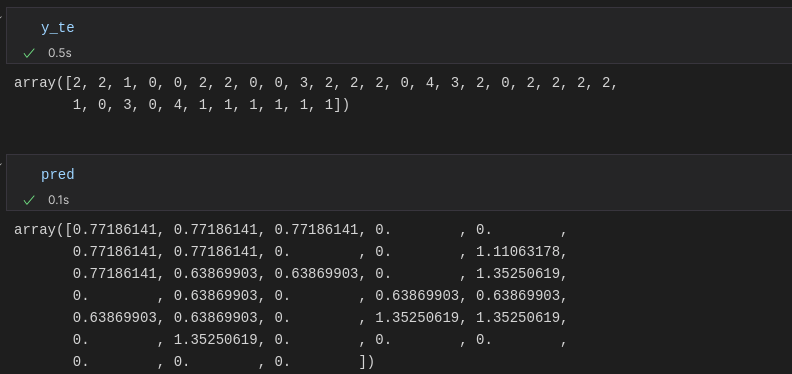
Once the data is prepared, we will train the "ADARANK" model, using the training features (X), ranking labels (Y) and query ids.



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

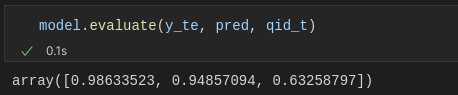


True values (manual ranking of the documents used for testing)



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.



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).

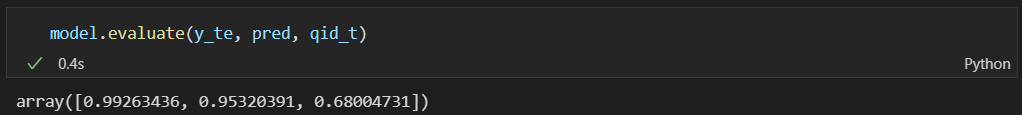
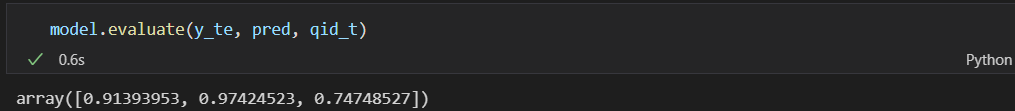
We tried to obtain a more realistic result by increasing the size of the dataset. We added 195 documents from LOINC to the dataset, prioritizing documents that would get a high value (2,3,4) according to the manual ranking method we previously explained. Our main goal was both to enlarge the dataset 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: Results of the enlarged dataset:

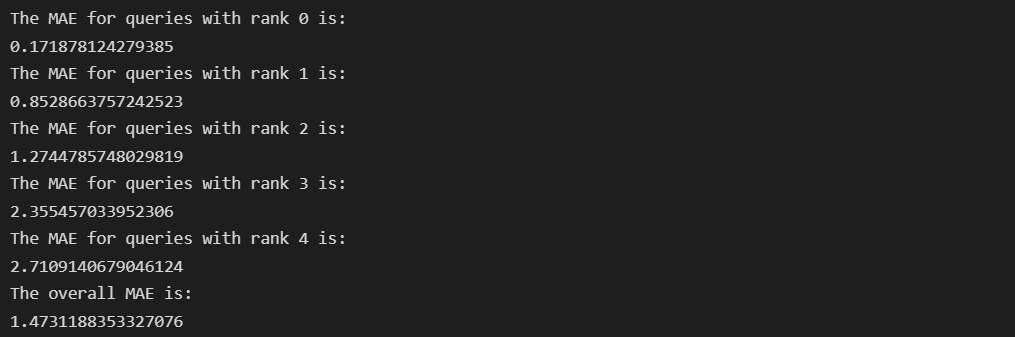
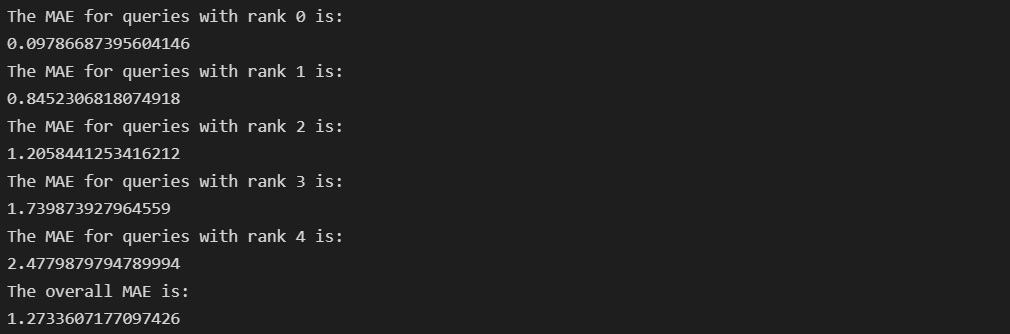
We can observe how the increase in size of the dataset has clearly improved the results of predictions regarding the third query, while having small negative variances in the other two.

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:  Results of the increased dataset:

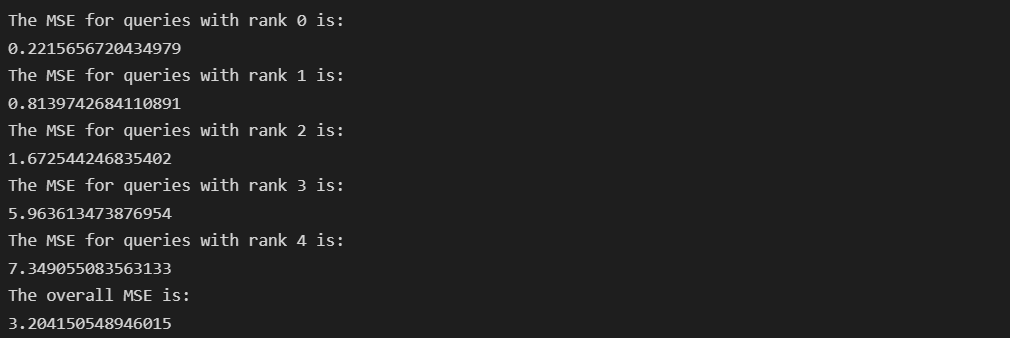
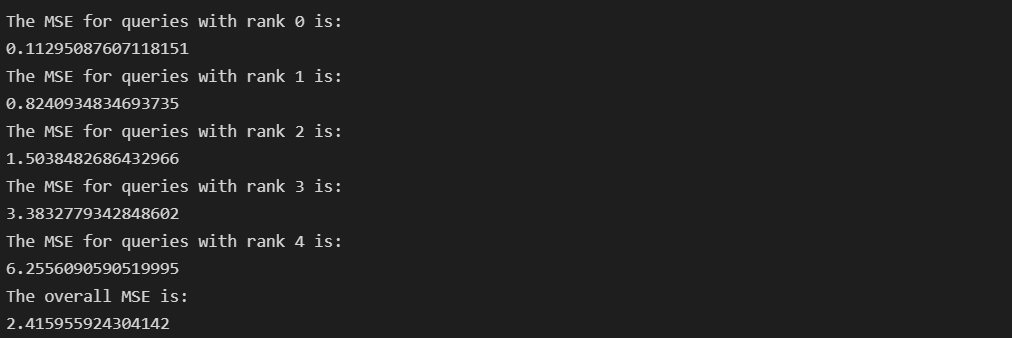
 

The goal of the analysis of this metric is to observe and demonstrate how increasing the dataset has improved the improved the ranking in terms of numerical difference between predicted and real values.

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:  Results of the enlarged dataset:

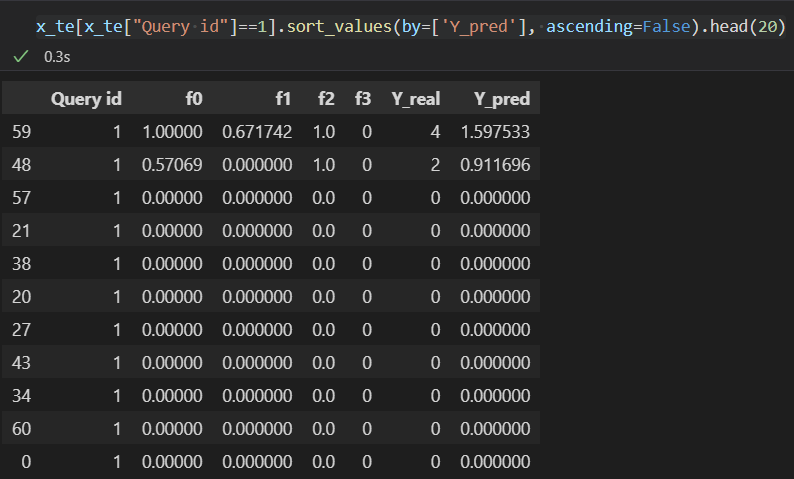
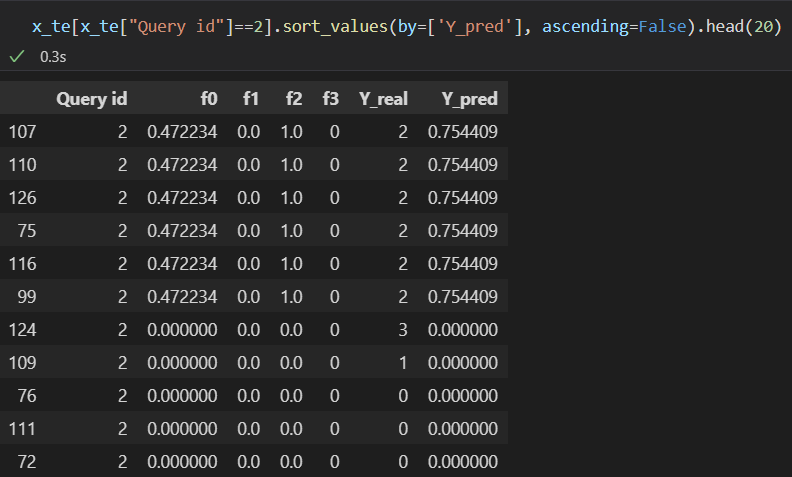
Similar to the previous metric, in this case we also want to analyze whether the new dataset has improved the ranking in terms of numerical difference between predicted and real values. MSE increases the relevance of the numerical difference because it works with square values, so it’ll allow us to see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

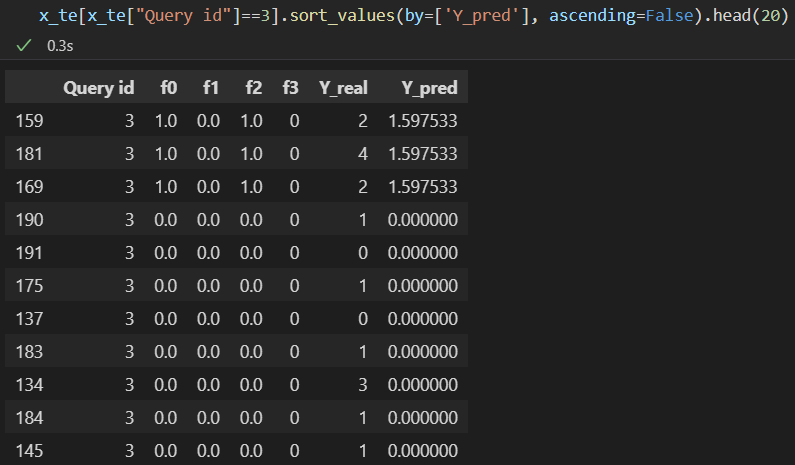
Thanks to that we can see more clearly how the increase of the dataset has had a relevant impact on higher ranked queries, improving a lot the results for values 3 and 4, and having a decreasing improvement over values 2, 1 and 0.

# 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):



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

