Máster Universitario en Inteligencia Artificial

AdaRank: Creating a ML based model for document ranking

ML ranking assignment – Biomedical Informatics

Authors:

Urtzi Sotes Senosiain

Imanol Solano Martinez

Professor:

David Pérez del Rey

November of 2020

Summary

1. Introduction

Machine Learning models are a type of mathematical models that can recognize certain type of patterns. The learning process is done by training a model over a set of data, it learns patterns that are extracted from the training subset. During the training, the actual best weights are validated with the validation subset. Finally, once the model has been fully trained, the model is tested on how good is at generalizing and making predictions for new data, this testing data will be called the test subset. It should be noted that an optimal training is based on the dataset, because after all the model learns patterns from the data and with them will be able to obtain its own conclusions. For this problem, we have been given a dataset that contain three different type of queries:

1. Glucose in blood.
2. Bilirubin in plasma.
3. White blood cells count.

Each of the queries contains a list of documents which will have a degree of similarity with respect to the query. Each query has 67 rows or 67 different documents, moreover, the file has 5 columns that correspond to different characteristics of the document. Below we will explain each of the characteristics to be able to get to the idea in greater depth:

- LOINC *number* → Identification number of the document in the LOINC dataset
- *Long common name* → Long name that best represents the document or article
- *Component* → Primary part of the LOINC name.
- *System* → Kind of quantity that is being measured in the test.
- *Property* → Specimen type or "unit of analysis" upon which the observation was made.

Each query of the dataset is represented as an array of shape 67 rows x 5 columns. The previous step to the training is a good analysis of the dataset to be able to understand it perfectly and therefore we have seen necessary the explanations above.

2. Training process

AdaRank is a boosting algorithm for information retrieval that has been based on the AdaBoost classification algorithm. AdaRank creates and uses a linear combination of weak rankers as its model. The main objective of the adaRank model is to develop a new learning algorithm that can directly evaluate the performance of the measure of an Information Retrieval system.

Information Retrieval Systems have the capacity of ranking documents of a query. Depending on what the document contains, the system computes a relevance score and with all the relevance scores of the documents, it will return a ranked list of documents in descending order. The larger the score value the more relevance the document will have within the query.
The objective of the training process is to construct a ranking function which will achieve the best results when ranking the training data in the sense of minimizing the loss function. Once we have analyzed theoretically the algorithm, we will get deeper on how we will train the model using the LOINC dataset.
First, the input features from where the algorithm will learn are going to be the long common name, component, system, and property. The long common name feature gives a lot of

information about the document as it summarizes all its content. The rest of the input features do not give as much information as the long common name, but they are also quite important and will improve the learning of the algorithm.

The objective is to build an algorithm or model that ranks documents as optimally as possible, therefore it does not make much sense to have the input variables as strings. Actually, the four input variables are strings that describe different characteristics of the document, so we will transform them into numeric values.

For sorting the documents within the *long common name* feature from top to bottom, we have followed the following steps:

1. Apply tf-idf model over all the strings. Tf-idf model is a metric that compares the value that a document has in the long common name feature with the query and returns a relevance score.
2. Considering the synonyms of the query words and the relevance scores obtained by the tf-idf model, make a ranking from 1 to total number of documents. Document number 1 will be the most relevant document on the feature long common name. It is very important to consider the synonyms because the tf-idf model gives a high relevance score to documents that contain words of the queries, but it can happen that they contain a synonym word, so the model will not appreciate it. For example, plasma and blood refer to the same concept, in the query of glucose in blood the tf-idf model will return a high relevance score for the document that contains blood and a low relevance score for the document that contains plasma.
3. Finally, we have a column with the relevance positions of the documents where number 1 refers to the most relevant one and number total documents to the less one.

The feature *component* is a word such as, Blood, Glucose, Chloride, Bilirubin, etc. These words appear repeated in different documents, this means that some documents will have the same ranking as others. For the ranking process we have followed the process described above, also taking into account the importance of synonyms. After making the ranking, in the three queries we have obtained documents ranked in the *component* feature from 1 to about 15.

The feature *system* contains an acronym of the quantity that is being measured in the test and again we have repeated words such as bld, Ser/Plas, Ser, Urine, Isolate, etc. In this case, the ranking of the documents has been done in a more intuitive manner because the acronym does not give much information. For example, if the query contains blood then documents with bld and Plas will be ranked before documents that contain words like Isolate or Dose because we do not know if those words have a lot to do with the query.

The feature *property* is the unit of analysis upon which the analysis has been made, it is the most scientific characteristic of the set and as the two ones before different documents contain repeated words. We have done an intuitive ranking and obtained again ranks from 1 to 15 in the three queries.

Finally, in each query we have added two new columns that are the query id (qid) and the real output or the ground truth (*label_ranking*). The query id will be 1 for all the documents in glucose in blood, 2 in bilirubin in plasma and 3 in white blood cells count. The real output is needed because the training of the adaRank model is supervised. The real output is a column with documents ranked from 1 to total number of documents. To be able to make a ranking of all the documents of each query, we have considered the four characteristics described above. In order to do this, we have given each document a position manually, according to how we believe our model should rank the documents knowing the values they have in each feature. It

should be noted that the most important feature for us is the *long common name* and that is why we have been guided by the most when making the slot that should predict our model.

At this point the data has been preprocessed correctly and each query is loaded into an array. The model "philosophy" for ranking is that the higher the value is in any of the features or in the actual output, more relevant the document will be. In our dataset it happens just the opposite since in position 1 we have the most important document and in position *n* (being *n* the last document) we have the least important document. Therefore, each column of the array is normalized to values from 0 to 1 and inverted as follows:

- The maximum value of the column is computed (the highest value).
- If we denote as x_i each element of the column, we transform it by applying this formula:

$$1 - \frac{x_i}{maximum\ value}$$

- We apply this method to the four input features and the real output columns of the three arrays.

Once the input data has been normalized and inverted documents that have high weights in the features will be relevant, we concatenate the three arrays into one. The new array will be a 201 x 6 array where the first four columns are the input feature, the fifth column is the query id, and the sixth columns is the real output. We have denoted the variables as we can see in Figure 1 as follows:

- As X = values of all the 201 documents in the four features columns.
- As y = values of all the 201 documents in the real output column.
- As qid = query id of all the 201 documents in the qid column.

```
X = final_data[:,:4]
y = final_data[:,5].ravel()
qid = final_data[:,4].ravel()
```

Figure 1: Python code for loading the variables.

The objective of the learning process is going to be to construct a ranking function that will achieve the best capacity of generalizing (test results). The initial dataset will be used to construct the three subsets needed for the learning and testing process: Training set that will be used for the learning process, validation set that will be used for validating the best weights distribution over the queries in the training data that has been returned after the learning process and the test set that will be used how good our model is when predicting new data.

For the first training experiment, we have divided the initial dataset into the training set and the test set taking the 80 % of the samples for the training subset and the 20 % of the samples for the test subset. We have found a problem when dividing it because it is essential to take separately the samples that belong to each query and from there divide them following the proportion 80, 20. Once they have been divided separately, we have concatenated the three train sets into one and the three test sets into a single. We have chosen to establish the training set as the validation set as well and with all this, we get an accuracy on the test set of the 64.04% and an accuracy on the training set of 82.65 %. The result on the test set might seem quite low but it is because the training set is too small for making a good real training.

```python
def split_samples(X, y, qid, query_id):
    indx = np.where(qid==query_id)
    num_samples = X[indx].shape[0]
    X_train = X[indx][:round(num_samples * 0.8),:]
    X_test = X[indx][round(num_samples * 0.8):,:]
    y_train = y[indx][:round(num_samples * 0.8)]
    y_test = y[indx][round(num_samples * 0.8):]
    qid_train = qid[indx][:round(num_samples * 0.8)]
    qid_test = qid[indx][round(num_samples * 0.8):]
    return X_train, X_test, y_train, y_test, qid_train, qid_test
```

```python
def concatenate_samples(X, y, qid):
    X_train1, X_test1, y_train1, y_test1, qid_train1, qid_test1 = split_samples(X, y, qid, 1)
    X_train2, X_test2, y_train2, y_test2, qid_train2, qid_test2 = split_samples(X, y, qid, 2)
    X_train3, X_test3, y_train3, y_test3, qid_train3, qid_test3 = split_samples(X, y, qid, 3)
    X_train = np.concatenate((X_train1, X_train2, X_train3), axis=0)
    X_test = np.concatenate((X_test1, X_test2, X_test3), axis = 0)
    y_train = np.concatenate((y_train1, y_train2, y_train3), axis=0)
    y_test = np.concatenate((y_test1, y_test2, y_test3), axis = 0)
    qid_train = np.concatenate((qid_train1, qid_train2, qid_train3), axis=0)
    qid_test = np.concatenate((qid_test1, qid_test2, qid_test3), axis = 0)
    return X_train, X_test, y_train, y_test, qid_train, qid_test
```

Figure 2: Python code for the division of the initial set into two subsets (train and test).

For the second training experiment, we have chosen to take the train and test sets built above but from the validation set we have taken out the validation set again taking the 80 % of the samples for the new training set and the 20 % of the samples for the validation set. For making the division of the training set into two new subsets we have used again the two functions shown in Figure 2. After training and evaluating the model with two different subsets we have tested it in the test subset, and we have obtained an accuracy of the 64,24 % and over the training set an accuracy of 82.65 %. As we can see, the results are practically the same as in the first training experiment we have done.

Finally, we have made another experiment training and validating the model with the whole dataset and then testing the model again with the whole dataset, and we get an accuracy of the 77.76 %. The model is learning patterns from the training data so when we are testing the model with the same learning data we are getting better results than in the two previous experiments because the model has been fitted on that data.

3. Conclusions

On the one hand, although at the start when the input data for the algorithm is preprocessed by making rankings from 1 to N, it is essential to normalize and invert those values by columns in order to convert them to the format in which the algorithm will accept them. The algorithm in the learning process associates that a document is very relevant if its relevance score in the feature it is been analyzed is very high.

On the other hand, the accuracy scores we have obtained in the three training experiments are too low because the dataset is very small, we will need more data to obtain better results. If we want to train optimally the AdaRank model we will need a dataset much bigger and wider which contains many more documents, features and it will have to be as varied as possible. If we are able to obtain a dataset of such characteristics, we will be able to train the model optimally because it will be able to learn more patterns and will be better at generalizing than the one we have developed in this work.