POLITÉCNICA

REPORT:
MACHINE LEARNED RANKING ASSIGNMENT

Information Retrieval Assignment 3

Ángel Igareta
David Burrell
Miguel Pérez
Rodrigo Pueblas

March 24, 2020

1 Domain

For this assignment, we were given a dataset of medical results from the LOINC database (Logical Observation Identifiers Names and Codes), which is a public standard for identifying medical laboratory observations.

The application domain will be build a model capable of ranking these results for any set of given queries, specifically:

- Glucose in blood
- White blood cells count
- Bilirubin in plasma

1.1 Selected Approach

For this purpose, we were given three different approaches to choose from: Pointwise, Pairwise and Listwise. We decided to go with the Pairwise approach, based on the paper "Optimizing Search Engines using Clickthrough Data" by Thorsten Joachims.

In this paper, the author presents a method that utilizes clickthrough data for training, connecting the query-log of the search engine with the log of links the users clicked on in the presented ranking. The advantage of this method in the web context is the huge amount of cheap and easily accessible click data available for training.

In this context, this data can be represented as a triplet (q,r,c) consisting of a query q, with a ranking of results presented to the user r and a set of entries that the user clicked on c. To encode this information, each query and link is given a unique ID, which is encoded along with the URL. It is important to note that this data is not absolute, as it is dependent on how much the user scrolled to find the results. Then, if the user clicked on the links ranked 1, 3 and 7 for instance, we can infer that the 7th link is more relevant than the links 2, 4, 5 and 6, but we do not know if it is more relevant than the links from 8th onward. Unfortunately, this kind of information does not suit well to standard machine learning approaches.

So instead the method employs the use of SVMs to include historical user data as part of a set of feature ranking results generated by different features. This gets around the issue of classification as the documents are not classified into a class, and also regression as no document has a defined score because it is user defined.

In order to define an objective function to maximize, the paper defines for an unknown distribution Pr(q,r*) of queries and target rankings on a document collection D with m documents a retrieval function for which Kendall's is the following:

$$\tau P(f) = \int \tau(r_{f(q)}, r^*) dPr(q, r^*)$$

It is clear now that in order to learn we have to implement a solution that maximizes the previous function, but we do not know yet which method is this...

2 Proposed solution

We want to be able to define a ranking score for each document. For this purpose, we can use the Cosine Similarity metric to compare the textual columns (long_common_name, component and system) with the query, resulting in determining how similar the query is to the text on the results irrespective of their size. This method measures the cosine of the angle between two vectors projected in a multi-dimensional space, representing the array of words in both of the documents being compared.

To be able to use these cosine results to rank the documents, each feature result is projected onto a weight vector to decide the ordering for the documents. Although this leads to an issue where each weight vector will have a different ordering for the documents. To find the optimal solution that meets all the weight orderings an SVM can be used to find the best overall ranking for the documents. To allow for further optimisation the SVM can be trained using data generated by users who have selected what they believe to be the optimum ordering of documents, or at least what they see as the subset of most important ones.

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

3 Dataset generation

As we do not have access to any real data, we had to assign a number of clicks for each result per query. To generate this data a python script was developed to create files that would act as user logs. This script used a collection of keywords foreach query and compared them to the long name value in the dataset. The keywords were made up of not just the words in the query but also some synonyms, for example for query 1 the words were:

"white", "blood", "cells", "count", "hemoglobin", "plasma", "leucocyte"

For each matching word a matching score was increased by 1. To add a random factor the final score was increased by a number between 2 and 6. Once the scores had been generated the documents were ordered in descending order by score (the highest scoring document first and the lowest last). This serves only as the initial training set, and once our model is trained, we can compare what our model predicts to be the ranking with the actual clicks of the users, which will be again used to readjust our model.

4 Implementation

For the final implementation, we converted our textual features using the cosine method described above, generating a set of features per query. The labels associated with those features were the expected order we wanted to achieve from those features and query.

Once the train dataset was created, we transformed the dataset pairwise, converting the n-class ranking problem into a two-class classification problem. Note that all pairs are selected, apart from those that have an equal target value, achieving an array of balanced classes.

Once we finished with the transformation, we fed that data to train a Support Vector Machine (SVM), which performs pairwise ranking with an underlying LinearSVC model.

The full implementation of the generated dataset and our attempt to build the corresponding model can be found in the following Github repository:

`https://github.com/angeligareta/MLRanking/`

5 Final results

For the given queries, we obtained the following results:

	Glucose in blood	**Bilirubin in plasma**	**White blood cells count**
#1	Glucose [Moles/volume] in Urine	Bilirubin total [Mass/volume] in Synovial fluid	Billrubin total [Presence] in Unspecfied specimen
#2	Glucose [Moles/volume] in Pleural fluid	Bilirubin indirect [Mass/volume] in Serum or Plasma	Nitrofurantoin [Susceptibility]
#3	Glucose [Moles/volume] in Serum or Plasma	Bilirubin direct [Mass/volume] in Serum or Plasma	Cholesterol [Mass/volume] in Serum or Plasma
#4	Glucose [Mass/volume] in Serum Plasma or Blood	Bilirubin total [Mass/volume] in Serum or Plasma	Trimethoprim+Sulfamethoxazole [Susceptibility]
#5	Cholesterol in l-iDL [Mass/volume] in Serum or Plasma	Cholesterol in l-iDL [Mass/volume] in Serum or Plasma	Blood group antibody screen [Presence] in Serum or Plasma

As can be seen from the results the final solution seems to have performed well on the first two queries. With the top 4 results for both returning titles that start with the same word. Although our method seems not to detect the rest of the sentence well. Unfortunately for the third query the results are poor with none of the top 5 results matching the inputted text.

6 Conclusion

The domain of ranking search result entries is very broad and complex. In some cases, such as internet links or medical data we can rely on the criterion of the user to make more relevant the results that are most clicked on. The huge advantage of this approach in this era is the amount of easily accessible, cheap data to support these algorithms.

On the other hand, we discovered how complicated it can get to implement these algorithms, even if they are not very complex in theory. Future work should be performed in order to address the last part of the implementation and finish the full working system.

As a first attempt we believe these to be reasonable results. Especially due to the team's lack of knowledge with these techniques and domain.

7 References

"Cosine Similarity." Wikipedia, Wikimedia Foundation, 12 Feb. 2020, https://en.wikipedia.org/wiki/Cosine_similarity
"Home." LOINC, https://loinc.org/
Joachims, Thorsten. "Optimizing search engines using clickthrough data." Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002.
"Pairwise ranking using scikit-learn LinearSVC", Github https://gist.github.com/fabianp/2020955