Information Retrieval - ML Ranking Assignment - 03/2024

ML Ranking (LOINC)

Universidad Politécnica de Madrid

Federico Paschetta, Cecilia Peccolo, Nicola Maria D'Angelo

federico.paschetta@alumnos.upm.es
cecilia.peccolo@alumnos.upm.es
nicolamaria.dangelo@alumnos.upm.es

Contents

Chapter 1

Introduction

In the dynamic realm of health data analysis, the application of machine learning techniques to decipher and forecast patient outcomes based on lab results is gaining paramount importance. This study concentrates on employing textual data manipulation and similarity evaluations to derive meaningful interpretations from lab tests standardized under the LOINC system. We conducts a structured analysis in Python, involving the preparation, mapping, and reduction of medical datasets related to blood glucose levels, bilirubin levels in plasma, and white blood cell counts. Key steps in the analysis include reading and preparing datasets, mapping data attributes for clarity and consistency, and applying a ranking calculation to assess or compare measurements across different datasets. This report summarizes the objectives, methodologies, and conclusions.

Chapter 2

Methodology

2.1 Creation of the training set

For each query, we aimed to create the training dataset using a binary judgment approach. Therefore, for each query, it was manually decided whether a document was relevant or not, based on the following criteria:

- it is relevant if the title and the components contain all the words of the query,
- it is relevant if the title and the components contain at least one word from the query,
- it is relevant if the title and the components contain a synonym and at least one word from the query.

This way, the result for each query will be a dataset with a list of documents classified in a binary manner as relevant or not.

2.2 ML ranking

To create a ranking of the documents, a ***point-wise approach*** based on a *vector space model* was used. The objective of this phase was to assign a rank value to each document, and then retrieve those with the highest scores, and then verifying their relevance based on the previous human coding. The ranking of documents was done using the **TF-IDF** of words present in the query without stop-words, document by document. Each score, called c_{ij}, was calculated as:

$$c_{ij} = f_{ij} \log \frac{N}{n_j} \tag{2.1}$$

where f_{ij} is the frequency of the j-th word in the i-th document and n_j is the number of document where j appears in all the N documents of the collection.
After computing the TF-IDF scores for the *"long common name"* and *"component"* parts of each document, we proceeded to compare the TF-IDF vectors of the documents and queries. This

comparison involved evaluating the proximity of these vectors using cosine similarity, calculated as follows:

$$similarity(x, y) = \cos(\theta_{x,y}) = \frac{xy}{||x||\ ||y||} \tag{2.2}$$

where x is the query tfidf score vector and y is the document tfidf score vectore calculated on the "long common name" and "component" fields, while $||x||$, $||y||$ denote their respective Euclidean norms.
Upon completing this meticulous process, we set a low threshold for each query, in order to consider as *relevant* only the documents with similarity to the query vector significantly different from 0 (which means that the two vectors are orthogonal, therefore not similar at all). These are the confusion matrix as following outcomes for each query:

TFIDF ranking/Labeled	Relevant	Non relevant
Relevant	44	0
Non relevant	0	23

Table 2.1: Confusionion matrix query "glucose in blood"

TFIDF ranking/Labeled	Relevant	Non relevant
Relevant	60	0
Non relevant	1	6

Table 2.2: Confusionion matrix query "bilirubin in plasma"

TFIDF ranking/Labeled	Relevant	Non relevant
Relevant	49	4
Non relevant	0	14

Table 2.3: Confusionion matrix query "White blood cells count"

2.3 Data set expansion

Comparing the results obtained manually with those obtained automatically by assigning a score to each document using TF-IDF, excellent results were achieved. Therefore, it was decided to expand the dataset, resulting in one comprising 102465 documents, in order to create bigger train set. On the other hand, the creation of the training dataset in this case is significantly complicated, since manual coding of each document for the queries would be too challenging. However, since the method of automatic document judgment yielded very good results, a labeled training dataset was created using the same methodology as in the previous phase, meaning by the use of TF-IDF scores. This dataset will serve as the target variable for future applications of other machine learning models for retrieving the most relevant documents.

Chapter 3

Implementation

3.1 Resources

In order to create training dataset, extend original one and develop the model it was necessary to start from a base of data and knowledge that could be useful for the process. As programming language it was used Python (in particular a Jupyter notebook) and were imported *pandas* and *re* libraries and, from *sklearn* library, *TfIdfVectorizer* module and *cosine_similarity* function. For the data a total of four datasets have been used:

- loinc_dataset-v2.xlsx, database got from Moodle, containing 3 pages, one for each query given, and 70 rows per page, with loinc_num, long_common_name, component and property attributes;
- LoincTableCore.csv, database downloaded from LOINC website, containing more than 100.000 rows, one for each LOINC data and many attributes;
- loinc_property.csv, list got from LOINC User Guide website, representing each meaning for each abbreviation in LOINC property attributes;
- loinc_system.csv, list got from LOINC User Guide website, representing each meaning for each abbreviation in LOINC system attributes.

3.2 Development

In the initial steps, each page in the Moodle Excel dataset was processed, where rows were already labeled with 1s or 0s. The Pandas function *read_excel* was leveraged to transform the data into Pandas DataFrames. This procedure was replicated for the entire LOINC dataset, excluding all LOINC fields not present in the Moodle dataset.

Subsequently, the remaining two CSV files were imported, and their information was stored in two dictionaries: *property_dict* and *system_dict*. These dictionaries were structured for easy access in subsequent operations. Using this information, property and system data in each DataFrame were mapped to their respective meaningful labels, ensuring interpretability for both humans and machines.

Next, for each dataset corresponding to the pages of the Moodle Excel file, scores were computed by invoking the *calculate_ranking* function. This function accepts the DataFrame and the desired query as parameters, handles the removal of any stopwords (in this case, a regular expression based on the 'in' stopword was constructed), builds a vocabulary from the query words, and creates a *TfIdfVectorizer* object using this vocabulary. Iterating over each row in the dataset, a TF-IDF matrix was generated, serving as the basis for calculating the ranking score via the *cosine_similarity* function. Subsequently, the 'score' column was appended to the final DataFrame, containing the computed values. Finally, this DataFrame was exported to an Excel file within the 'training' folder.

Following a thorough analysis of the results, the process was replicated for the extended dataset, generating scores for the provided queries across all LOINC data. The resulting DataFrames were also written to the same folder for further examination and utilization.

Chapter 4

Conclusion

After preliminary analysis and implementation, we have a folder with the following structure:

```
|   LoincTableCore.csv -> full dataset LOINC
|   loinc_dataset-v2.xlsx -> starting dataset (with labels 1 and 0s)
|   loinc_property.csv -> Property attribute map
|   loinc_system.csv -> System attribute map
|   mlranking.ipynb -> main notebook code
|   mlranking.py -> notebook code translated to native Python
\---training
        bilirubin_in_plasma.xlsx -> ranking of all LOINC codes (with more than 0.0 score)
        breast_cancer.xlsx -> ranking of all LOINC codes (with more than 0.0 score)
        glucose_in_blood.xlsx -> ranking of all LOINC codes (with more than 0.0 score)
        white_blood_cells_count.xlsx -> ranking of all LOINC codes (with more than 0.0 score)
        init_bilirubin_in_plasma.xlsx -> training set with our label and code prediction
        init_glucose_in_blood.xlsx -> training set with our label and code prediction
        init_white_blood_cells_count.xlsx -> training set with our label and code prediction
```

Looking at the predicted scores of our implementation compared to manually labeled binary relevance scores (both visible in training datasets beginning with *init*), we observe that our model performs well in ranking based on term occurrences. However, it struggles when handling synonyms or technical terms. For instance, in the query "glucose in blood," we can easily split rows as actually relevant or not using any threshold, as the rows with zero relevance receive scores of zero from the model. Conversely, in the query "white blood cell count," the model performs poorly. For example, it fails to recognize that terms like "leukocytes" and "white blood cells" are synonymous: surely the addition of Natural Language Processing modules could enhance its performance.