iscte TECNOLOGIAS E ARQUITETURA

Algoritmos para Big Data
Lab class handout #7

Recommendation system **2024/2025**

Learning objectives

This lab class is about creating and using a recommendation system for books, with particular interest on the collaborative filtering strategy.

Note: It is acceptable that the work to be undertaken may require some time beyond the class. If that is the case, work should continue outside of class. And if needed, assistance will be provided during the office hours established or to be agreed with the lecturer.

Supporting information

- Course slides
- Apache Spark MLlib – Collaborative Filtering
- YData Profiling

Introduction

A recommendation system tries to establish a kind of relationship between people and items. There are various algorithmic strategies to rely upon, namely collaborative filtering and latent factors.

In this lab class, we will be creating and then using a recommendation system of books. The working data relates to data initially collected by Cai-Nicolas Ziegler in a 4-week crawl from the *Book-Crossing* community, with some processing afterwards e.g. deleting columns with no relevant interest for the problem or with poor data quality. Despite the improvement made, one can not assume that the provided data is 100% free of errors. The archive, containing three tables of information – users, books, and ratings – can be downloaded from:
`https://bigdata.iscte-iul.eu/datasets/books-recommendation.zip`

Finally, alongside this handout describing the tasks to accomplish, a notebook is provided for the implementation of some tasks.

Tasks to accomplish

A. Data ingestion, data preparation and understanding

In this initial task, execute the following operations:

1. Download the archive/data files mentioned above and save them in a proper directory, within the working directory of the Pyspark instalation.
2. Create a notebook specifically for the implementation of this task. As a suggestion, look at a notebook from the lab class handout #5, in the part related to data profiling, and create the notebook from there.
3. After creating a *SparkSession* to work with, read each dataset into dataframes, let us call them `df_users`, `df_books` and `df_ratings` respectively. Check the dataframes.
4. With the tool YData Profiling, generate profile reports concerning each of the above dataframes. Do not require information regarding correlations among columns. Save each report as an html file and then, through a browser, open each file and analyze the generated profile.
5. Carry out operations over the dataframes so we may guarantee that data will be prepared/cleaned to be worked with. Hence:
   (a) Analyze in detail the profile report that was generated.
   (b) If needed and/or advisable, carry out cleaning operations over the dataframes.
   (c) Next, make sure that any new and transformed dataframe – the outcome of the cleaning operations just mentioned – is saved into a parquet file, with a proper name.
6. Regardless of changes being made or not, for each clean dataframe create another one as a 30% sampling (Feel free to use a different sampling fraction, but substantially lower than the maximum).

   As before, saved them as parquet files, with proper names that should include the suffix `_small`.

   This concludes the first notebook. The tasks hereafter will be implemented in the notebook provided.

B. Data supporting the model

In this task we will setup the working data. At this point, it will be from the smaller data size version. Hence, execute the following operations:

1. After creating a *SparkSession* to work with, read into dataframes the parquet files of smaller size saved in the previous task. Let us call then `df_users`, `df_books` and `df_ratings` respectively.
2. Check in general the data quality of the dataframes that were just created (leave outliers aside).

C. ML recommendation model

Turning now to the recommendation model, the relevant dataframe to work with is `df_ratings`. Furthermore, we will follow the usual procedures to create a ML pipeline, like we did in the *Lab class handout #4 – Machine learning problem – binary classification*. Hence, execute the following operations:

1. Looking at feature engineering, from the three key features of interest to build the model – User-ID, ISBN and Book-Rating – we conclude that ISBN has to be transformed since it is of string type and ML algorithms only work with numbers. Therefore:

   (a) Set a transformer stage with *StringIndexer* to convert *ISBN* to a new feature *ISBN-Index* of numeric value.
   (b) Specify the features to be considered for creating the model.

2. Create two dataframes – `df_train` and `df_validation` – after `df_book_ratings`, randomly, with distinct sizes. For instance, use a split of 70%, 30% respectively.

   Also, save these dataframes as parquet files. Recall again that, as it is a sampling, there is no guarantee of getting the same data split when using the code in a different computer/time.

3. Just for the purpose of having more memory space hereafter, delete `df_book_ratings` as we no longer need it.

4. Set the estimator as *ALS*, which works as a recommendation algorithm. See details in the Spark documentation about Collaborative Filtering.

5. Set a ML pipeline configuration using the transformer *Pipeline*, with the two stages leading to the model. That is, related to *StringIndexer()* and *ALS*. Save the pipeline for further use, should it be required.

6. Create the model by fitting the pipeline that has been set up to the training data, `df_train`. Save the model for further use, should it be required.

7. Evaluate the model that has been created. Specifically:
   (a) Make predictions by applying the validation data, `df_validation`, to the model (which is a transformer).
   (b) Print out the schema and content of the resulting dataframe.
   (c) Compute the evaluation metric *rmse* with the help of the evaluator *RegressionEvaluator*.

D. Model deployment

In this task, we aim to get recommendations upon the model. Fortunately an *ALS* model allow us to get recommendations directly, so we will take advantage of that. It means we no longer follow the normal procedure of getting predictions using the pipeline set. Hence, with the help of functions associated with the *ALS* model, show with as much information as possible the following:

1. The top 2 book recommendations for a specified subset of users, for example, 3 users who have smaller IDs.

2. The top 2 user recommendations for a specified subset of books, for example, 3 books that have smaller indexed IDs. (Such users might be interested on the specified books.)

Notice that, in order to follow the strategy mentioned, there is a need to redo the model training without pipeline.

E. Additional exercise

As an additional exercise, try to further extend the capabilities of both model creation and deployment by:

1. Exploring how the running model can sustain higher numbers for top recommendations (instead of just 2). Likewise for all users and books.

2. Using the bigger data size version to create the model.