Building an Information Search System

Information Processing and Retrieval Project

Paulo Ribeiro
up201806505@edu.fe.up.pt
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

Pedro Ferreira
up201806506@edu.fe.up.pt
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

Pedro Ponte
up201809694@edu.fe.up.pt
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

Abstract

This report pretends to detail the implementation of an information search system for the curricular unit **Information Processing and Retrieval**, for the Master in Informatics and Computing Engineering. This implementation was divided into three milestones, relative to data collection and preparation, information querying and retrieval, and retrieval evaluation.

Keywords: datasets, data processing, information retrieval, retrieval evaluation, information search system

1 Introduction

The goals of the first milestone of this project are to correctly prepare the data for the information retrieval tool to be implemented in the second milestone. The first step is to explore different datasets and choose a captivating one. After being familiar with its data, the next step is to think of a possible data processing pipeline, adequate for our problem.

Upon defining this pipeline, some essential steps are necessary, such as assessing the data quality by looking for missing or invalid values and handling them, besides fixing some other minor problems. In our case, data extraction was also applied to fill or update missing values. A step focused on data exploration is also required, with the gathering of some important statistics and the creation of distinct plots representing the data involved.

2 Milestone 1 - Data Preparation

2.1 Data Collection

For this step, our first move was choosing an interesting collection of data. This meant picking a dataset with a somewhat large amount of information that would be useful and relevant for the given purpose.

While seeking a dataset worthy of fitting the requirements, various options that ended up not being worked upon came across. Datasets related to videogames, as a gathering of all players present in the EA sports franchise FIFA or club and sportsman data from SEGA game Football Manager, were some considered examples. The digital distribution service Steam, from Valve corporation, was also taken into account. Although these datasets were able to fill almost all the requirements needed, the lack of sizeable texts of individual information could be a non so good aspect that would limit us in the future.

So, the final choice ended up related to the topic of Netflix's TVSeries and movies. Two different datasets about this topic were found. The choice on which one to go with was based on the existence of a vaster number of relevant parameters and the presence of a summary column that would be useful for full-text searches needed for future work.

The information was also gathered from the IMDb website. This would facilitate the search for missing intel and bring confidence to the work while using a trustworthy source for the data collected since IMDb is a well-known and prestigious platform for online TV series and films.

The dataset chosen has information in the form of the following columns:

- *imdb_id*: Show id from the IMDb website;
- *title*: The title of the show;

- *popular_rank*: The popular rank of the show;
- *certificate*: The age restrinction of a show;
- *startYear*: Release year of a show;
- *endYear*: Year the show ended;
- *episodes*: Number of episodes of a show;
- *runtime*: Mean time duration of a show;
- *type*: Type of a show;
- *orign_country*: Origin country of the show;
- *title*: The title of the show;
- *language*: The language a show was produced on;
- *summary*: Summary of the show;
- *rating*: IMDb rating of the show;
- *numVotes*: Number of votes for show rating;
- *genres*: Genres categories of a show;
- *isAdult*: Presence of adult content in the show;
- *cast*: Cast of the show;
- *image_url*: Poster of the show.

2.2 Pipeline Design

After deciding which dataset to explore, our next step is to define a concise pipeline for the data processing, so that we structure its steps in the most adequate way to our problem. We started by documenting it using a diagram to better visualize the data flow. This diagram is shown in the following figure:

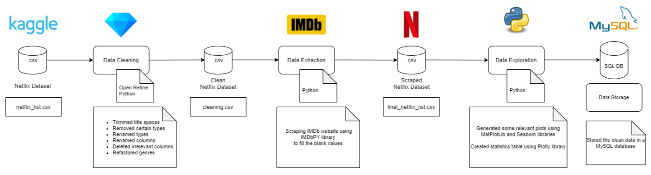


Figure 1. Data processing pipeline

The starting point of our pipeline consists of the Netflix dataset, which we retrieved from Kaggle, in a CSV format.

Then, our first step is to clean this data by fixing some issues and adapting it to our best interest. This includes removing some irrelevant columns from our dataset, renaming others and refactoring some data formats to make them more easily accessible. The previous step results in a clean CSV file, where these changes are already applied.

After this, we thought it was interesting to fill the missing values of the dataset with recent information gathered from the web. The indicated step is called Data Extraction. It was decided to resort to the IMDb website and apply some scraping to complete the information about the shows since it is the world's most popular and authoritative source for this type of content. The fact that the original dataset was built retrieving information from this same source also helped in the decision. Since we desired a consistent dataset, the newly collected data shouldn't be too different from the existing one.

The aforementioned process results in the last CSV file, from which it can now be performed the Data Exploration step, where we gather some important statistics of our problem and generate some relevant plots to better illustrate some of its characteristics.

Finally, our last step is to store the data in a way that its access becomes easy and efficient. For this, we decided to use a MySQL database. After all these steps, the data is now ready to be retrieved every time it is needed.

2.3 Data Cleaning

Initially, we have a dataset with 7008 rows and 19 columns. Some of the columns contain irrelevant information, others have repeated data and some of them contain similar categories that could be joined together.

In our analysis on *OpenRefine*, we conclude that the column *isAdult* adds no value to our dataset, since it has all values as '0', thus we removed it. We also checked that the columns *plot* and *summary* are very similar, but the *summary* has more information than *plot*, so we opted to delete the *plot* column.

Then, we noticed there were some pairs of repeated titles, although one has a leading whitespace whereas the other doesn't. We found out that the ones with no initial space represent a 'tvSeries' and the others were a 'tvEpisode'. After some search on IMDb platform using the *imdbID*, it was concluded the 'tvEpisodes' are, in reality, episodes of the 'tvSerie' with the same name, so we opted to delete all the rows that have 'tvEpisode' as *type*.

Continuing *type* analysis, the dataset only has one entry with 'videogame'. Since we consider that only having one value of this type doesn't add any value to our future search system, we decided to drop this entry. There are also two rows that, in addition to having a null *type*, do not have other several values defined, so they were also taken off. After removing these types, we still have: 'tvSeries', 'tvMiniSeries', 'tvShort', 'tvSpecial', 'tvMovie', 'video', 'movie', 'short'. We decided to rename 'tvSeries' for 'series', 'tvMiniSeries' for 'miniSeries', 'tvShort' for 'short', 'tvSpecial' for 'special', 'tvMovie' for 'movie' and 'video' for 'animation'.

Finally, it was decided to rename some of the columns in order to have them all in *lowerCamelCase* style and, for each *genres* column entry, put all its values inside an array.

Finished the cleaning step, we have a dataset with 6216 rows and 17 columns.

2.4 Data Extraction

Once the data cleaning process is finished, we proceed to an attempt to fill in the missing values.

To do this, we used Python and a library called IMDbPY. With this library, we can get almost all types of information about the IMDb shows.

In the case of *certificate* column, initially we had only a certificate for each show, but the country to which the certificate applies was not always the same. Taking this into account, we decided to extract from IMDb all the certificates that apply to each show and store them into a dictionary converted into string, in the format of *country : certificate*.

To try to complete some of the missing values in *startYear* and *endYear* columns, we filled all the shows that are missing both values simultaneously and all the series or miniSeries that don't have the *endYear* value.

To complete the null values in *episodes* column, we have scraped from IMDb the number of episodes of each series or miniSeries that are missing this value.

For *runtime*, *originCountry*, *language*, *summary*, *rating*, *numVotes*, *genres* and *cast* columns, we try to extract from IMDb the missing values of these dataset parameters.

2.5 Data Exploration

Now that our data is finally clean and complete, we are ready to start exploring all its 6216 entries. In this step, we generate some relevant graphics so that we can better visualize our information domain and possibly deduce some conclusions. We used Python to achieve these results, namely the **MatPlotLib** and **Seaborn** libraries to generate the charts and the library **Plotly** to create a statistics table.

Our first thought was to get an overview of one of the most important attributes of our dataset: the type of the show. We decided that a Bar Chart was the most suited approach to see the dimensions of each of those types. The graphic is shown in the next figure:

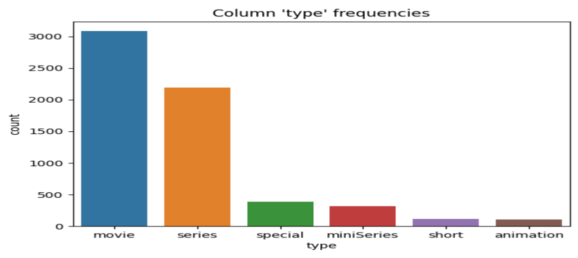


Figure 2. 'Type' Bar Chart

As we can see, the dataset is especially filled with movies and series, followed by some less frequent types, such as specials, mini-series, shorts and animations.

Then, we thought it was interesting to study the evolution of the ratings over the decades. For this, we decided to use a Box Plot, since it gives us the most information about that distribution, namely the ranges, outliers and quartiles. Before generating the graphic, we first had to group the years by decades, exclude the "Not available" values and sort the decades. That resulted in this figure:

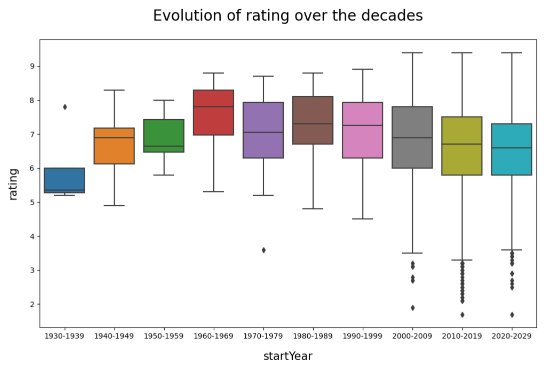


Figure 3. 'Rating' Box Plot

The number of shows from recent decades is naturally higher than the number of older shows. As a consequence, the rating range of the recent decades is larger. Both the minimum and maximum of the ratings were achieved in the 21st century. The mean of the ratings has also been slightly decreasing throughout the years. It's also possible to notice some outliers, marked by the dots.

Next, we wanted to see the distribution of the languages and origin countries. We knew there were too many distinct values in these columns to represent them all in a single diagram. Thus, the best and cleaner way we found to solve this problem was to create a Pie Chart, where we would call one of the divisions "Others". This chart should show the nine most common languages/origin countries and the *Others* division would encompass all the less frequent ones. Both of these charts can be seen in the following figures:

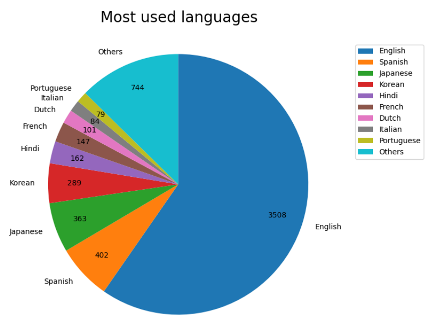


Figure 4. 'Language' Distribution Pie Chart

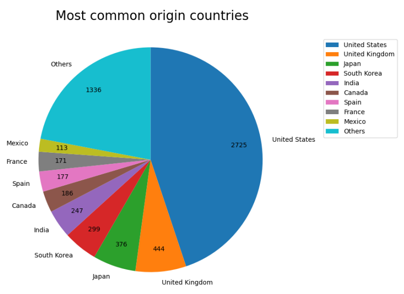


Figure 5. ’Countries’ Distribution Pie Chart

We also thought it was interesting to show the distribution of the genres. This was a more challenging quest since each show can have many genres associated. So we had to create and fill a dictionary where for each genre there is a count associated, with the number of times that genre is present in a show. Then, we generated a Bar Chart with the nine most common genres and the *Others* column encompassing the less frequent ones, which is shown in the following figure:

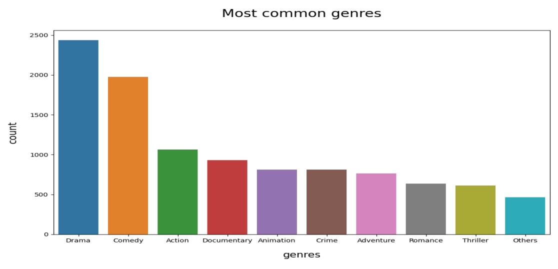


Figure 6. ’Genre’ Bar Chart

Similarly, it would be relevant to know the actors with the most appearances in our dataset. The same strategy of the genre chart was applied, and we got a dictionary containing every actor and their number of appearances. After this, we generated the chart with the top 20 of the actors with the most participations, which can be seen in the next figure:

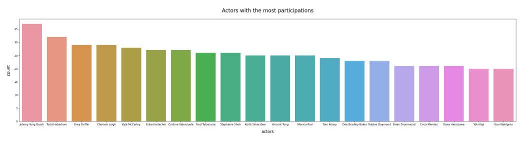
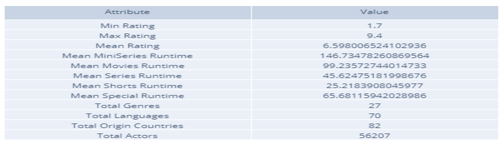


Figure 7. ’Cast’ Bar Chart

Finally, we wanted to know some basic statistics of our dataset, such as the means of the runtimes of each show type and the total number of distinct values of some columns, which we gathered in a table, shown in the following figure:



Attribute	Value
Min Rating	1.7
Max Rating	9.4
Mean Rating	6.598006524102936
Mean MiniSeries Runtime	146.73478260869564
Mean Movies Runtime	99.23572744014733
Mean Series Runtime	45.62475181998676
Mean Shorts Runtime	25.2183908045977
Mean Special Runtime	65.68115942028986
Total Genres	27
Total Languages	70
Total Origin Countries	82
Total Actors	56207

Figure 8. Basic statistics of the dataset

2.6 UML Design

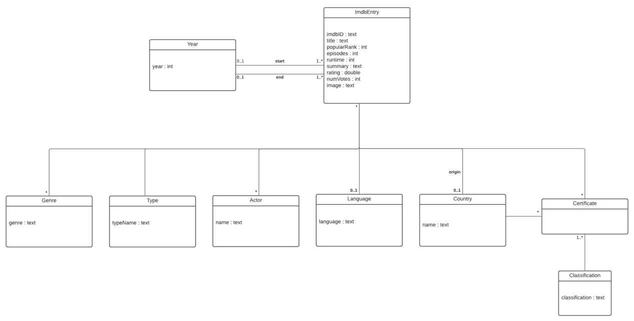


Figure 9. UML diagram for database creation

To be able to create a database that could support saving all the information in a useful and efficient way, a UML was carefully thought and created.

The main class ImdbEntry was created to represent each show. This class is composed by a parameter imdbID, title, popularRank, episodes, runtime, rating, summary, numVote and image.

As there are two columns dedicated to years on our dataset and repeated values for many shows, a new class Year was created having two different relations with ImdbEntry (one for the startYear and the other for the endYear).

Furthermore, five new classes were created since the same issue as in the year appeared, where the values would be very repetitive. This way, a class genre (with the genres a show can have), type (class with the various types a show can be, whether it is a tv series, film, etc), actor (class with the name of an actor that can participate in a show), language (languages a show can be originally made on) and country (countries where the show can be taken place) were created.

Finally, a new class Certificate was created, as well as a connection to the ImdbEntry. This object also has a connection to the country class and to a new classification class that will give the actual value for the certification. This new class classification will depend on the country correspondent. This happens because different countries classify their shows on different scales and will be useful in future work, allowing us to get search results in all the different classifications.