Wikipedia Movies Search System

Julián Ferreira
Tiago Antunes
up202202340@fe.up.pt
up201805327@edu.fc.up.pt
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

ABSTRACT

This paper provides an analysis of the process that has been carried out in order to develop a search system for movies. In order to achieve this, a dataset retrieved from Kaggle with different information about movies from Wikipedia was used.

The following task was to clean and prepare the data for future tasks involving a pipeline to carry out this process. After, a brief analysis of the data was conducted by generating different charts and graphics to have a better understanding of our dataset in a more visual way.

Once this was finished it was time to develop the information retrieval system. This includes the indexing of different attributes and the retrieval of information with queries.

Finally, we improved our search system by adding more fields to our dataset, adding some more filters to the plot text, and developing a tool that recommends some documents similar to the ones given by the search.

KEYWORDS

datasets, data, movies, indexes, queries

1 INTRODUCTION

Nowadays all information can be turned into data and consequently stored in a database. From the user's point of view, the large amount of data can sometimes be confusing and difficult to understand. The goal of this report is to deal with a dataset in a way that makes it easy to visualize. For this purpose, we selected a dataset of movies.

The report is divided into three different sections. The first one documents all the data preparation process. This includes the choice of the dataset, its content, the evaluation of its quality and the liability of the source, the pipeline used to carry out the data preparation, the different aspects that have been made in the data refinement, and finally the analysis of the data, including different charts and tables to extract the maximum information of our dataset. The goals of this milestone are to prepare the data for it to be used in the following parts of the project, to obtain the maximum possible information on our data, and to set the prospective search tasks for the next milestone.

The second part is the information retrieval. This starts with the election of the information retrieval tool, which in our case has been Solr. This is followed by the Collection and indexing of our dataset with the description of the chosen schema. After is the retrieval and evaluation of our data. This contains different queries that have been processed and evaluated in order to obtain some metrics of our dataset to reach the conclusions presented at the end of this report. The main objectives of this section are to satisfy the prospective search tasks from the previous milestone and to evaluate the queries with the chosen schema and boosts.

The third and last section is dedicated to improving our final search system. The main objective is to add more functionality and refinement to our system and queries, respectively, in order to upgrade the user experience and to retrieve better results.

2 M1 - DATA PREPATATION

"Data preparation is the process of gathering, combining, structuring and organizing data so it can be used in business intelligence (BI), analytics and data visualization applications." [5]

2.1 Dataset Choice

The chosen dataset was not the first option we thought of. The initial idea was to work with sports data, such as football statistics, but then we realized that it was quite hard to find this type of dataset with rich text, so we ended up changing our minds. The next option is the one we ended up working with, a movie dataset.

First of all, we thought of IMDB. The problem we had with this is that it was not possible to get a free API license, and the free datasets that were available did not have the text we were looking for. After this, we started searching in Kaggle [6].

This is a dataset with around 35,000 movies from Wikipedia, which is quite a trustable source. We searched for some extra information and we saw it is updated regularly and is valued at 8.82 in usability by Kaggle [4].

2.2 Dataset Content

We have a unique dataset with 34.893 rows that contain the following information:

- `Release year`: The year in which the movie was released
- `Title`: The title of the movie in English
- `Origin/Ethnicity`: The origin of the movie
- `Director`: The name, or names, of the director of the movie
- `Cast`: The names of the main actors
- `Genre`: The genre the movie is qualified as
- `Wiki page`: The link to the page of the Wikipedia of that movie
- `Plot`: A brief summary of the movie

2.3 Data Quality and Source

Kaggle is a well-known site for data analysts and the dataset is very documented. It has a ranking of 8.82 in usability so we considered it a trustable source. To analyze the quality of the data we did a random search of some movies on the internet to check that our set was correct.

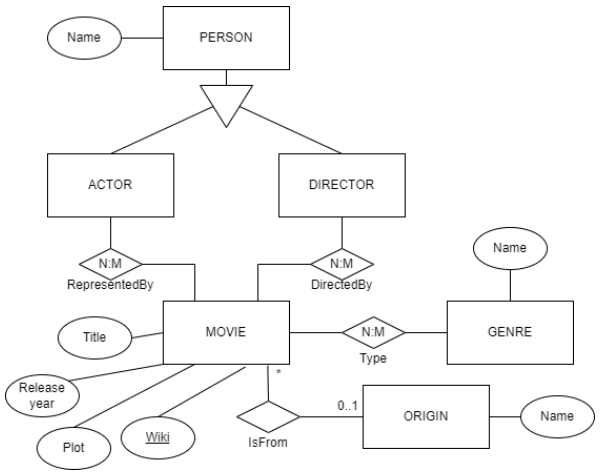

Figure 1: UML of dataset created on Diagrams.net [1]

A problem we do have is that there is some missing data. This results in having some empty fields or with the word "unknown". We considered that it is not a huge problem for our work because it is sometimes irrelevant, and it is normal that some information is missing in such old movies.

2.4 Pipeline

Our Pipeline is built almost entirely on python scripts. With the help of *pandas* library, we handle and manipulate the data using simple scripts to clean and organize the data.

We considered it was not necessary for us to put our dataset in a SQL database system due to that it was more comfortable to work with the csv file we already had. By doing this, we have avoided the time and work it would have taken us to do a database.

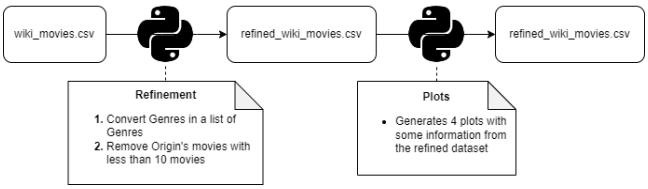

Figure 2: UML of the pipeline created on Diagrams.net

Figure 2 summarizes how we obtained, from our initial dataset, multiple plots with some relevant information about the dataset.

2.5 Data Refinement

When it comes to data refinement, the first thing we did was convert the `Genre` column to a list of genres. Imagine three rows of the original dataset:

- `Id`: 1 | ... | `Genre`: drama
- `Id`: 2 | ... | `Genre`: romantic
- `Id`: 3 | ... | `Genre`: drama romantic

If we wanted to know how many different genres are in the example above, the return would be 3 and not 2 as it should be. So we created a little script on python that transform the example above into:

- `Id`: 1 | ... | `Genre`: [drama]
- `Id`: 2 | ... | `Genre`: [romantic]
- `Id`: 3 | ... | `Genre`: [drama, romantic]

This script is not 100% efficacious and it has some flaws. The main idea of this script is to split the Genre string every time it finds a white space. South African, James Bond, and Warner Bros are examples of genres with two words and the script sees them as two different genres. To solve these circumstances, after the script split the two words, we check if match one of the cases and concatenate again those two words.

The second and last thing we did about data refinement was to remove the origin with less than ten movies. In our dataset, we had two origins - Assamese and Maldivian - and both had less than ten films. So we removed exactly eleven movies (nine Assamese and two Maldivian) from our dataset.

2.6 Data Analysis

In order to obtain more information about the dataset we have developed a python script with several different functions. Each one of these tries to extract some characterization of the data we are handling and represent the results of the exploration with diagrams or relevant data.

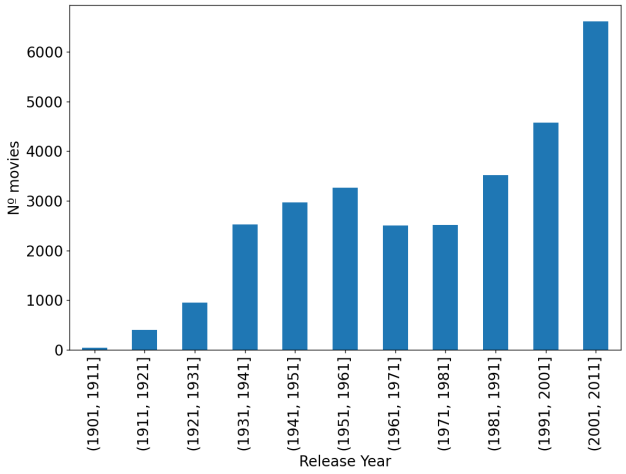

Figure 3: Movies produced per decade

Figure 3 represents the number of movies that have been produced in each decade. It is easy to tell that our dataset has the most movies in the 2000s decade, which is quite normal because these are the most recent. Another thing that we could appreciate is that the dataset has more or less the same number of movies from the 1930s to the 1980s because the objective of this dataset is to have information on older movies as well.

Figure 4 represents the different movies grouped by their origins. In order to represent this we decided to set a minimum number of movies for that origin to appear in the resulting diagram so that it could be easier to understand it. All these genres that did not reach the minimum number are classified as "Others".

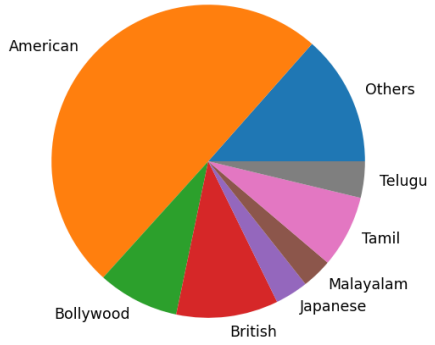

Figure 4: Movies grouped by origins

About the results obtained, it was easy to predict that most of the movies are of American origin but what was unexpected is that some other ethnicities that are not so known or common also have a high percentage such as Tamil or Bollywood. In conclusion, we can establish that this dataset's objective is to have information with high diversity, even though it might be from not-so-known movies.

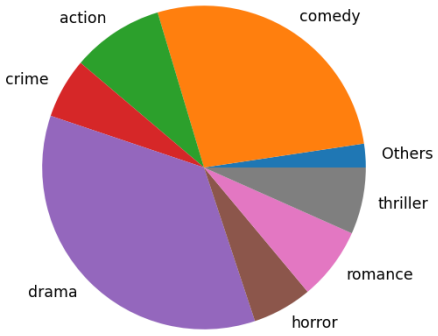

Figure 5: Movies grouped by genres

Figure 5 is the result of grouping the movies by their genres, the same way as figure 4. For this we did exactly the same as before, establishing a minimum number of movies for a genre to appear in the diagram. By analyzing the result, we can tell that the most common genres are drama and comedy by far different from the rest of the genres.

Table 1 represents the keywords and their frequency in the plot of the movies in our dataset. At first, there was no minimum length for these keywords but the result was irrelevant because all the words we found were straightforward articles that are repeated a lot. So we decided to increase the length and after a couple of different tries, we decided to set it as words with more than 5 characters. By doing this, we found much more conclusive results.

The 10 most frequent words in the plots represent the most common themes or expressions in order to do a summary of the movie. It is interesting that three of these words are related to family, which gives us the idea that most of the plots are related to it. Also, the word police is very self-explanatory because it represents the high number of movies related to crime in our dataset.

Keyword	Frequency
before	13,998
father	13,055
police	12,166
family	11,260
becomes	10,254
decides	10,140
through	9,808
himself	9,447
However	9,440
mother	9,057

Table 1: Keywords of movie's plots

2.7 Prospective Search Tasks

Given the nature of our dataset, we want to be able to take advantage of most of our attributes when searching for a movie. In order to do that, we have the possibility to make multiple search tasks such as:

- Search for a movie by its Title
- Search for a movie by its Gender
- Search for a movie by its Origin/Ethnicity
- Search for a movie by its Release Year (with ranges)
- Search for a movie by its Director
- Search for a movie by its Cast (Actors)
- Search for a movie by keywords in the Plot

It is important not only to allow single attribute searches but also to combine them, for instance, if we want to know which movies were directed by Steven Spielberg from 1990 to 2005, this can be done in our dataset.

3 M2 - INFORMATION RETRIEVAL

"Information retrieval is finding material (usually documents) of an unstructured nature that satisfies an information need from within large collections" [9]. And that's what we are going to talk about in the second part of this report.

3.1 Information Retrieval Tool + Collection

We decided to use Solr [8] to build our information retrieval system and to do that, with the help of the Solr reference guide [7], we ran Solr8.10 inside a docker [2] container. This software provides better tools for indexing and querying long textual fields and also supports CSV files as input.
Since we are going to work with just one dataset file, in Solr configuration, we only created one core to store the information.
After the docker image with the Solr is up and running, we populated the core.

3.2 Indexing

To know which fields we needed to index, we have taken a look at the prospective search tasks subsection and see which ones could be used later in the query. In addition, we have created a set of field

types and associated each field with a field type depending on its characteristics. In table 2, we can see the description of each field and its configurations.

Field	Type	Solr class	MultiValued
Title	synonym_text	TextField	false
Origin	porter_text	TextField	false
Plot	plot_text	TextField	false
Release Year	number	IntPointField	false
Genre	synonym	TextField	true
Director	char_text	TextField	true
Cast	char_text	TextField	true

Table 2: Description of schema fields

Every field type has one Solr class, an index analyzer, and a query analyzer. Each analyzer has a tokenizer, which in our case will be always the Standard Tokenizer from Solr, and a list of filters.

The *number* field type is the only one that is not a string, so for that case, we didn't apply any token or filter. Below, we will describe and explain each filter used on each field type with TextField has Solr class:

- **synonym_text**
  - ASCII Folding Filter: this filter converts alphabetic, numeric, and symbolic Unicode characters which are not in the Basic Latin Unicode block (the first 127 ASCII characters) to their ASCII equivalents, if one exists
  - Lower Case Filter: converts any uppercase letters in a token to the equivalent lowercase token. All other characters are left unchanged
  - Synonym Graph Filter: this filter maps single- or multi-token synonyms, producing a fully correct graph output
  - Porter Stem Filter: this filter applies the Porter Stemming Algorithm for English. The results are similar to using the Snowball Porter Stemmer with the language="English" argument
- **porter_text**
  - ASCII Folding Filter
  - Lower Case Filter
  - Porter Stem Filter
- **plot_text**
  - ASCII Folding Filter
  - Lower Case Filter
  - Synonym Graph Filter
  - Porter Stem Filter
  - English Possessive Filter: this filter removes singular possessives (trailing 's) from words
  - Hyphenated Words Filter - this filter reconstructs hyphenated words that have been tokenized as two tokens because of a line break or other intervening whitespace in the field test. If a token ends with a hyphen, it is joined with the following token and the hyphen is discarded
  - Stop Filter: this filter discards or stops analysis of, tokens that are on the given stop words list. A standard stop words list is included in the Solr conf directory, named stopwords.txt, which is appropriate for typical English language text
- **char_text**
  - ASCII Folding Filter
  - Lower Case Filter
  - Mapping Char Filter: this char filter is used for changing one string to another (for example, for normalizing é to e)

3.3 Retrieval and Evaluation

In this subsection, we have four different information needs built from the prospective search tasks and we will explain each one of them, as well as the query that we used and the results that we obtained. Since the number of retrieved documents wasn't high, we analyzed all documents to determine which ones were relevant. We used two systems to test the information needs: in the first one, we just used the indexed fields described in the previous subsection; in the second one, we applied different boosts depending on the fields that were more important in the information needs.

3.3.1 Evaluation metrics.

- **Average Precision** (AvP): value obtained for the set of documents existing after each relevant document is retrieved
- **Precision**: value obtained from the number of relevant documents retrieved divided by the number of retrieved items
- **Precision at 10** (P@10): value obtained from the number of recommended documents that are relevant divided by the number of recommended documents (10)
- **Recall**: value obtained from the number of relevant documents retrieved divided by the number of relevant documents

3.3.2 Queries.

- **Query 1**: Drama movies with Bollywood origin released in 2015 or after
  - **Relevance**: The relevant results are considered the ones that match all the attributes and their unique gender is drama.

Option	Value
q	Genre:drama AND Origin_Ethnicity:bollywood AND Released_Year:[2015 TO *]
qf (boost)	Genre^5 Origin_Ethnicity^3 Released_Year

Table 3: Arguments of query 1

  - **Observations**: As we can see in table 4 and figure 6, there is no improvement between both systems. The reason is that after analysing the results of the non boosted system, this are the best possible for this query, so they can not be improved.

Metric	Value	Value with boost
AvP	0.80	0.80
P@10	0.99	0.99

Table 4: Precision metrics for query 1

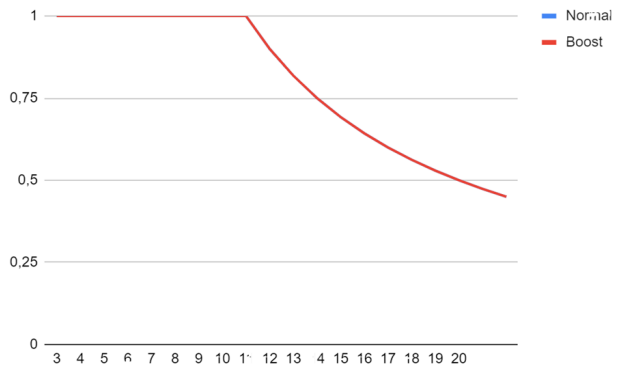

Figure 6: Precision recall curve of query 1

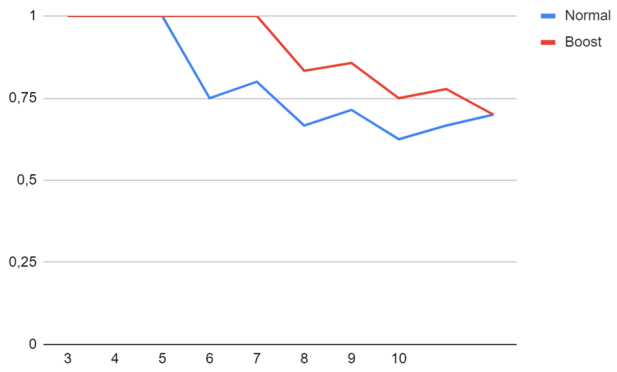

Figure 7: Precision recall curve of query 2

- **Query 2**: Movies starring Robert de Niro and are from the Italian mafia
  - **Relevance**: Movies that are considered relevant in this query are the ones that match have Robert de Niro as an actor and, after reading their plot, we can determine they are about italian mafia.

Option	Value
q	Cast:"Robert de Niro" AND (Plot:mafia OR Plot:Italian)
qf (boost)	Cast^5
bq (boost)	Plot:mafia^3 Plot:italian

Table 5: Arguments of query 2

  - **Observations**: Looking at the figure 7 and the table 6 we can conclude that there has been quite an improvement in the results after the boosts, mainly due to the field boost of the keyword "mafia" in the plot, because this word is much more important for the query than "italian".

Metric	Value	Value with boost
AvP	0.79	0.89
P@10	0.79	0.89

Table 6: Precision metrics for query 2

- **Query 3**: Movies directed by Steven Spielberg about murder
  - **Relevance**: Movies that are considered relevant in this query are the ones that are directed by Steven Spielberg and, after reading their plots, we can determine they are about murders(crimes).
  - **Observations**: As we can see in figure 8 and in table 8 this query has not been very precise, but still the boosts have improved its precision. The reason for the results in this query not being as good as in the others is that this is quite a complex query because there are many movies whose plots include words such as kill or murder and they are not about real murders. For instance, those words can appear as an explanation of the origin of the character.

Option	Value
q	Director:"Steven Spielberg" AND (Plot:murder OR Plot:kill)
qf (boost)	Director^5
bf (boost)	Plot:murder^3 Plot:kill

Table 7: Arguments of query 3

Metric	Value	Value with boost
AvP	0.50	0.57
P@10	0.61	0.70

Table 8: Precision metrics for query 3

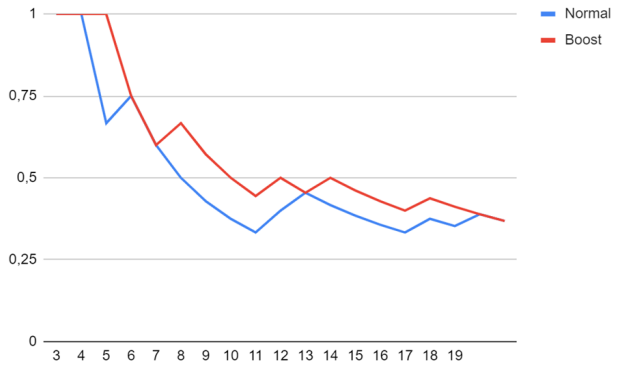

Figure 8: Precision recall curve of query 3

- **Query 4**: Police action movies from the year 2000
  - **Relevance**: The relevant movies considered in this query are the ones that match the gender and release year attributes and, after reading their plots, we can determine they are police movies.
  - **Observations**: Figure 9 and the table 10 show us the results of this query. In average precision this has been the worst query of the four of them, but after using the field

Option	Value
q	Plot:police AND Genre:action AND Release_Year:2000
qf (boost)	Plot^3 Genre^2 Release$_{Y}ear$
bf (boost)	Plot:police^5 Genre:action^3

Table 9: Arguments of query 4

boosts this metric has been triplicated. The reason the results are not really good is the same as in the previous query. There are many movies whose plot contains the word police which are not police movies. So it is very difficult to obtain relevant results with this query.

Metric	Value	Value with boost
AvP	0.39	0.43
P@10	0.44	0.49

Table 10: Precision metrics for query 4

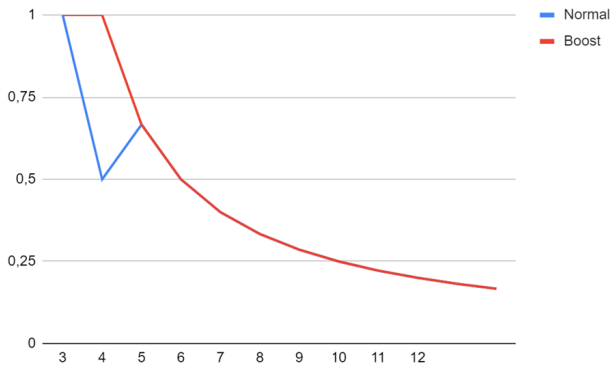

Figure 9: Precision recall curve of query 4

4 M3 - SEARCH SYSTEM

"Search systems are built to be used by humans, thus a search user interface is necessary for users to interact with the system" [9].

In our project, we didn't implement a user interface and stuck with Solr's interface. So, for this last part of the report, we are going to implement some improvements in our search system.

4.1 Improvements

Regarding the improvements to our search system, we made two things. The first one was adding a new field called rating to our dataset. To do that, we downloaded two IMDb datasets [3], one with the basic information about the movie and another one with the movie ratings, and after that, we merge our initial dataset with the IMDb one using the Title as the key.

Our second improvement, is we add a MoreLikeThis which is a tool from Solr that enables users to query for documents similar to a document in their result list. So, the idea is when we search for a movie, it will appear some movie suggestions that are similar to the ones that appear from the result of the query.

The third and last improvement is the implementation of a search system with "words related to a theme". This means that for a certain query, in the plot, it is not only searched the certain token and its synonyms as it was before. Now it also searches for words related to the one being searched. The way this has been implemented is with the Solr.SynonymGraphFilterFactory, where we already had a "synonyms.txt" file, which we didn't removing, we just added more lines to it with the similar words to a certain theme.

4.2 Evaluation and Discussion

In order to evaluate the movie recommender mentioned in the previous section, we have selected five movies from our dataset to test them and get the five best matches. These movies are famous and we are familiar with them, helping us to define the relevance of the results obtained.

These movies are the following: Batman, The Godfather, Shutter Island, and Gran Torino. Some of these movies are part of a saga, so we expect the recommender to match those that belong to that saga. However, other aspects are taken into mind at the time of suggestion.

The fields that are considered in order to suggest a certain movie include the title, the director, the cast, the genre, and the plot. To get a better evaluation, we used two different boosts for these fields, giving some of them more relevance. The boosts used are the following:

- **Boost 1**: `Title^6 Director^2 Cast^1 Genre^2 Plot^1`
- **Boost 2**: `Title^4 Director^3 Cast^2 Genre^3 Plot^1`

Being this stated, we can proceed with the results of the evaluation for each of the movies. * The plot is considered related if, after reading, both storylines somehow match.

4.2.1 *Batman.*

We can appreciate seeing in both tables (12 and 13) that the first boost suits much better what we are searching for. The reason is that it shows all the movies from the saga Batman, even though they do not share a director or cast.

The reason this changes with the second boost is that the importance of the title is reduced, resulting in the recommendation of some movies that have not got so much to do with the original. In these results the plot makes a huge impact because, even though their plots might not be correlated, there are some keywords repeated several times in the recommended movies causing these to have better values than others.

In conclusion, the first boost is exactly what we are looking for with this query.

4.2.2 *The Godfather.*

* Unique table due to both boosts giving the same result In this case, we have the exact same results with both boosts, so there is no possible comparison between these two.

After analyzing the results, these could be improved, but in the end, the most important movies (the others from The Godfather saga) are included, so it is quite what was expected. Here, the plot makes a big impact, the same as in the previous query.

Movie	Title	Director	Cast	Genre	Plot
Batman returns	Yes	No	No	Yes	Yes
Batman forever	Yes	No	No	Yes	Yes
Batman begins	Yes	No	No	Yes	Yes
The Lego Batman movie	Yes	No	No	Yes	Yes
Batman and Robin	Yes	No	No	No	Yes

Table 11: Batman results with Boost 1

Movie	Title	Director	Cast	Genre	Plot
Batman returns	Yes	No	No	Yes	Yes
The invisible Ray	No	No	No	No	No
Batman forever	Yes	No	No	Yes	
Dracula's Daughter	No	No	No	No	No
Batman begins	Yes	No	No	Yes	Yes

Table 12: Batman results with Boost 2

Movie	Title	Director	Cast	Genre	Plot
The Godfather Part II	Yes	Yes	Yes	Yes	Yes
The Godfather Part III	Yes	Yes	Yes	Yes	Yes
Jane Austen's Mafia!	No	No	No	No	Yes
Redline	No	No	No	No	Yes
Heavy Traffic	No	No	No	No	Yes

Table 13: The Godfather results with both Boosts

4.2.3 *Shutter Island.*

The recommended movies for each query do not really determine which of the boosts is best for this query, however both of them have very interesting results. The main difference between them is that in the first Boost the plot has a bigger impact, resulting in the recommendation of psycho-thriller movies, just as Shutter Island. Meanwhile, the second boost is more influenced by the cast and genre. Nevertheless it still gives, in most occasions, movies with similar storylines.

Being this stated, we can conclude that both boosts work properly for this query regardless the differences. It would all depend in the priority of the user.

Movie	Title	Director	Cast	Genre	Plot
The Ring	No	No	No	No	Yes
Captain Newman, M.D.	No	No	No	No	Yes
Elephant Song	No	No	No	Yes	Yes
Half Light	No	No	No	Yes	Yes
Ring 2	No	No	No	No	Yes

Table 14: Shutter Island results with Boost 1

Movie	Title	Director	Cast	Genre	Plot
The Mangler	No	No	Yes	No	Yes
The Ring	No	No	No	No	Yes
The Silence of the Lambs	No	No	Yes	Yes	Yes
Little Boy	No	No	Yes	No	No
Captain Newman, M.D.	No	No	No	No	Yes

Table 15: Shutter Island results with Boost 2

4.2.4 *Gran Torino.*

* Unique table due to both boosts giving the same result In this case, we have the exact same results with both boosts, so there is no possible comparison between these two.

It was known previously of the evaluation of the query that it would be quite difficult to obtain good results due to the complexity of this movie's plot. However, we still get some really interesting results such as Gangs of New York, which is also about wars between mafias, something similar to the storyline of Gran Torino.

Movie	Title	Director	Cast	Plot	
Assault on Precinct 13	No	No	No	No	Yes
In the Nick	No	No	No	No	Yes
Gangs of New York	No	No	No	Yes	Yes
The Constant Woman movie	No	No	No	Yes	No
The Virgin Suicides	No	No	No	Yes	No

Table 16: Gran Torino results with both Boosts

4.2.5 *Final conclusion of the recommender.*

The conclusion we come up with after the evaluation is that both boosts work properly in most cases, but the first one is much better when it comes to movies from sagas and the second takes more into account other fields like the genre, the director, and the cast. However both are too influenced by irrelevant words present in

the plot, but even though we have made several changes in the recommender we have not been able to control this.

4.2.6 Similar words. In order to test and try this improvement, we are going to use the fourth query from last milestone. Just to remember, this query is about police action movies from the year 2000.

The relevant movies are those ones that match the gender and release year and, after reading their plots, we can determine they are police movies. The arguments are shown in table 9.

Metric	Normal	Boost	Similar
AvP	0.39	0.43	0.64
P@10	0.44	0.49	0.69

Table 17: Precision metrics for query 4

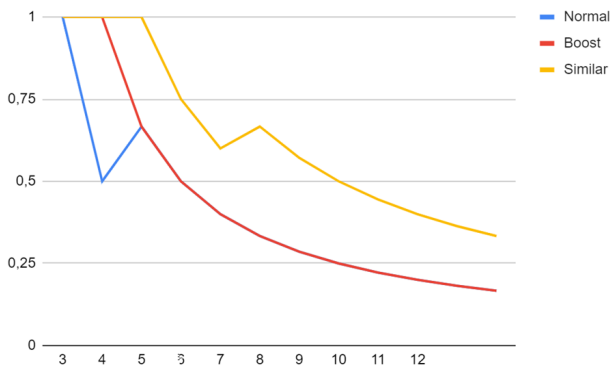

Figure 10: Precision recall curve of query 4

With this little improvement, the query has found two new relevant documents it didn't before, improving considerably the results as we can see in figure 10, where all the precision recall curves are shown for the different boosts.

5 CONCLUSION

All the objectives set in the first milestone were successfully accomplished. Despite having some difficulties with reading the CSV file. The satisfaction of the objectives has allowed us to have a better understanding of the dataset we are working with and this is being prepared for the next stages of the project.

Regarding the second milestone, we can consider we have made possible the prospective search tasks described in the first milestone so, we can conclude that the main objective has been accomplished. Apart from this, we have to be critical of the results of the evaluation that has been carried out. The values obtained in the different queries that were executed can be improved, but still, the usage of personalized boosts has helped a lot. In every single case, the boost chosen for the query has improved the result, so this objective has also been accomplished.

The main objectives of the third and last milestone were to improve the information retrieval and the user experience. On the one hand, we added a new field to the dataset, which is the average rating of each movie and we implemented a similar word search. We have considered that the average rating can be a really interesting attribute for the user in order to realize more interesting queries. And as figure 10 shows, the similar word search has been an improvement for those queries that do not search for a certain word, if not a certain theme.

On the other hand, we have implemented a movie recommender based on the class MoreLikeThis from Solr. This allows the user to, for a certain movie, get some suggestions about what movies are similar to that one. After testing and evaluating these improvements, mainly the recommender, we can reach the conclusion that it gives us a new functionality for the user, which can be very interesting, and that it works properly giving use quite good results for each query.

REFERENCES

[1] Diagrams.net. 2000. Flowchart Maker Online Diagram Software. Retrieved October, 2022 from https://app.diagrams.net/
[2] Docker. 2013. Docker: Accelerated, Containerized Application Development. Retrieved November, 2022 from https://www.docker.com
[3] IMDb. 2022. IMDb Datasets. Retrieved December, 2022 from https://www.imdb.com/interfaces/
[4] JUSTINR. 2018. Kaggle: Wikipedia Movie Plots. Retrieved September, 2022 from https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots
[5] JUSTINR. 2022. What is data preparation? Retrieved February, 2022 from https://www.techtarget.com/searchbusinessanalytics/definition/data-preparation
[6] Kaggle. 2010. Kaggle: Your Machine Learning and Data Science Community. Retrieved October, 2022 from https://www.kaggle.com/
[7] Solr. 2021. Apache Solr Reference Guide. Retrieved November, 2022 from https://solr.apache.org/guide/8_10/
[8] Solr. 2022. Apache Solr. Retrieved November, 2022 from https://solr.apache.org/
[9] Moodle UP. 2022. Moodle UP - Processamento e Recuperação de Informação. Retrieved December, 2022 from https://moodle.up.pt/mod/folder/view.php?id=48459