RecFlix - Netflix titles recommendations

1st André Flores
up201907001@fe.up.pt

2nd Guilherme Garrido
up201905407@fe.up.pt

3rd Luís Lucas
up201904624@fe.up.pt

***Abstract*—This document serves as a report for this project that deals with the analysis, processing, characterization, retrieval, and querying of a dataset of Netflix movies and TV shows. The goal of this project is to develop an information system that allows users to search for Netflix titles by multiple criteria such as genre, runtime, and description, and to provide recommendations of similar movies and shows. To achieve this, firstly, the dataset was analyzed, refined, and characterized. Then, using Solr, the data was indexed and retrieved according to the main information needs. Finally, the retrieval tasks were evaluated according to their precision and recall.**

I. Introduction

The streaming of movies and shows is increasingly in fashion, being one of the most common hobbies nowadays, regardless of age. The objective of this project is to develop an information search system, which includes work on data collection and preparation, information querying and retrieval, and retrieval evaluation on a Netflix movies/shows dataset. This dataset was chosen because it has both textual (title, description, genres) and numeric (runtime and scores) data. RecFlix is a movie/show search system that allows for the search of titles using many parameters and also includes recommendations helping users to find similar titles to the ones they know.

II. Data Preparation

The dataset used in this project refers to movies and shows available on Netflix streaming in July 2022, extracted by Victor Soeiro. It was obtained from Kaggle (License CC0: Public domain) and originally included two files: titles.csv and credits.csv, with 5850 and 77801 records, corresponding to 1,93MB and 3,63MB, respectively.

A. Data processing pipeline

Figure 1 illustrates the data processing pipeline. Firstly, analyzing the original data, using OpenRefine exhaustively, allowed the identification of inconsistent records, mostly in the titles.csv file. Some records were deleted: duplicates and records with missing data (described in the next section). Then, it was decided to split the data from the titles.csv, extracting the information related to ratings/scores and genres. Therefore, two new files were created: scores.csv and genres.csv. Finally, the data was explored and characterized.

B. Data operations

Inconsistent data were mostly found in the cleaned titles.csv file. Duplicate titles (1 record removed) and missing descriptions(18 records removed), records with no information about age certification (2619 records set to 'unrated' to not disturb statistics), and the movie's runtime declared as 0 (14 records replaced by 'null' to not disturb statistics) were the data issues found. These deletions led to the need to remove the connected rows in credits.csv, so actors and directors of the movies/shows deleted were removed as well as they have no value now.

Also, in the file created regarding the scores, some records were found without values related to the score, voting, and popularity on the IMDb and TMDB platforms. The information about TMDB could be fetched from its website as the movies/shows are identified by title.id which is available in every record. On the other hand, the IMDb system works with a different identifier (imdbId in the scores.csv) which is missing in 403 records. These records cannot have their IMDb scores and votes fetched so the values were left empty.

No more inconsistencies were found.

C. Final dataset

Title.csv keeps the information about the movie/show and has the following columns:

- id - movie/show identifier on JustWatch platform
- title - the title of the movie/show
- type - MOVIE or SHOW
- description - brief description of the movie/show
- release year - the year of release
- age certification - adequate age for the audience of the movie/show
- runtime - length of the movie or episode (show)
- production countries - list of countries that produced the movie/show
- seasons - number of seasons available for the show

Credits.csv keeps the information about the actors and directors of the movie/show:

- person id - person identifier on JustWatch platform
- id - movie/show identifier
- name - the name of the actor/director
- character - the name of the character the actor played
- role - ACTOR or DIRECTOR

Genres.csv columns:

- id - movie/show identifier
- List of all genres present in the original titles.csv. Each cell has 1 as value if the corresponding movie/show is considered from the corresponding genre and 0 otherwise.

Scores.csv columns:

- id - movie/show identifier on JustWatch platform
- imdb id - movie/show identifier on IMDb platform

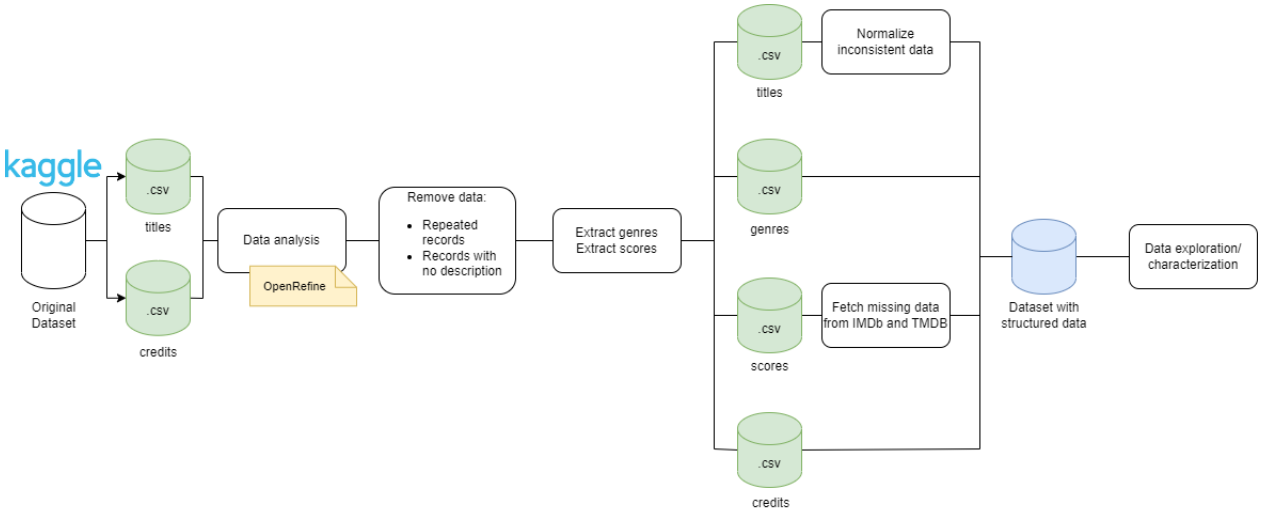

Fig. 1. Data processing pipeline.

- imbd score - movie/show score on IMDb platform
- imdb votes - movie/show votes on IMDb platform
- tmdb popularity - movie/show popularity on TMDB platform
- tmdb score - movie/show score on TMDB platform

D. Conceptual data model

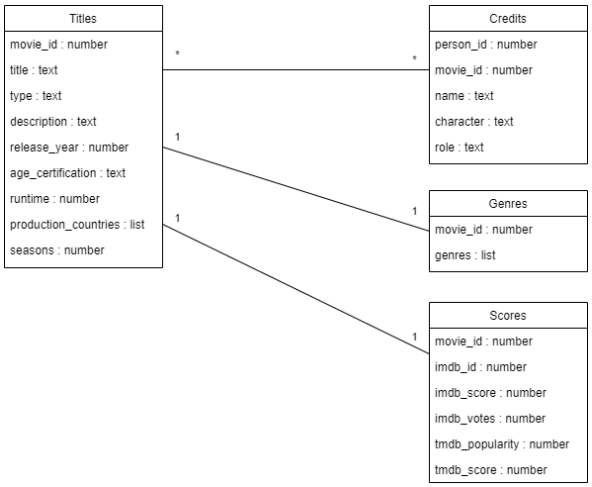

Fig. 2. Conceptual data model.

After analyzing the original data, it was decided to split the titles domain, which included genres and scores. So it ended up with four main domains (Figure 2): titles, credits, genres, and scores, each corresponding to a file. The attributes in the model are the columns of the respective data structure and already were described in the last section "Final dataset". A title can have multiple credits and an individual can be credited in multiple titles. A title has only one list of genres and each list is associated to just one title. A title has a set of scores and a set of scores is associated to only one title.

E. Makefile

The makefile produced follows the process of data operations shown in the pipeline. It starts by deleting the entries with duplicate titles or without descriptions. Then, it creates new files for genres and scores and deletes the respective columns from titles. Also alters records with no information about age certification, setting it to 'unrated', and records with 0 runtime, replacing it with 'null'. Furthermore, the makefile updates the records with missing scores by fetching them from the IMDb and TMDB websites. Python scripts were used to create the genre file and to fetch scores.

F. Data characterization

In this section, some histograms are presented to help understand the content of the project's dataset.

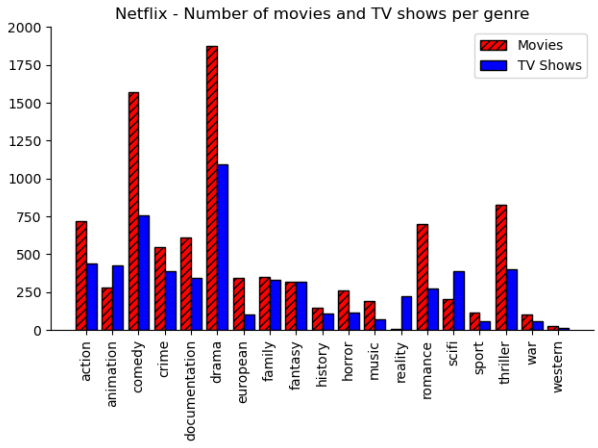

Fig. 3. Movies/shows distribution per genre.

The titles.csv file has almost 6000 registers. Figure 3 shows the distribution by genres and by type (movie or show). It is visible that drama, comedy, thriller, action, and romance

are the most common genres of movies/shows, and there are significantly more movies than shows.

The next histograms (Figures 4 and 5) show the evolution of the production of movies and shows. Only 3,5% of titles were produced before the year 2000 so it was decided not to include that data here.

The movies followed an exponential growth until 2017, peaking in 2018 and decreasing in the following years (Figure 4). The TV shows followed a similar evolution until 2018 but on a smaller scale, peaking only in 2021 (Figure 5).

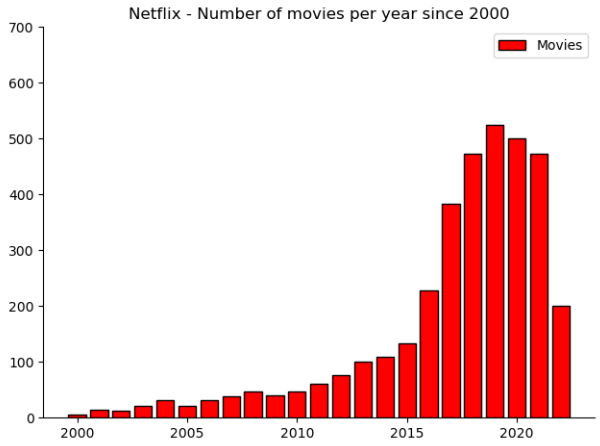

Fig. 4. Movies distribution per year since 2000.

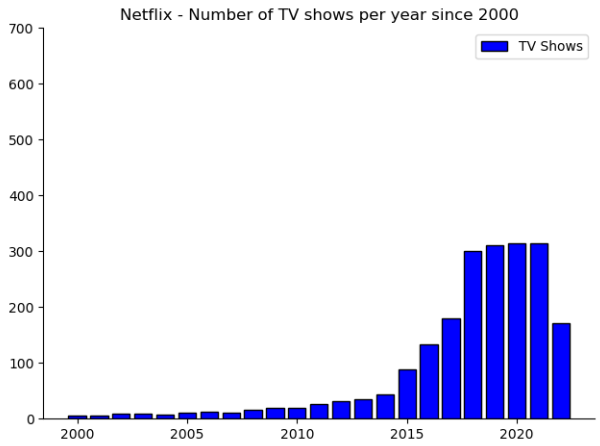

Fig. 5. Shows distribution per year since 2000.

Figures 6 and 7 show, in histograms, the average score, on the IMDb platform, of movies and shows, respectively, grouped by the year of production. It can be concluded that the audience tends to prefer older productions, as the scores given to the more recent movies are lower than the older ones, and they also prefer TV shows to movies.

Figure 8 shows the average scores of movies/shows grouped by genre. TV shows are usually better rated than movies from the same genre. Animation, european, horror, romance,

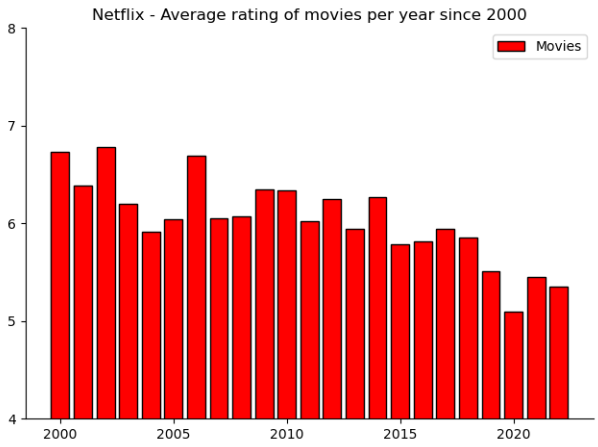

Fig. 6. Average movie score per year since 2000.

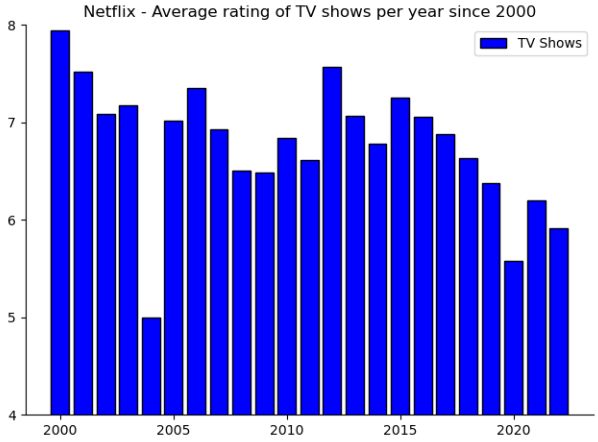

Fig. 7. Average TV show score per year since 2000.

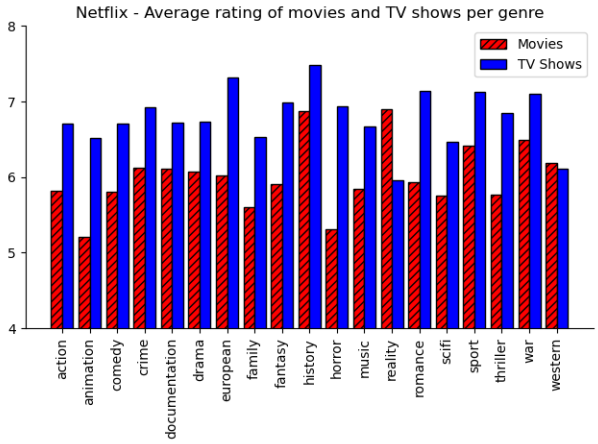

Fig. 8. Average movie/show score per genre.

and thriller have the biggest differences between movies and shows.

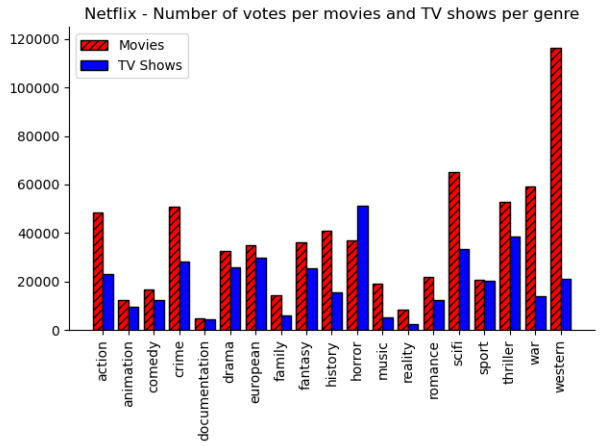

Fig. 9. Average votes on movie/show per genre.

Figure 9 illustrates the average number of votes per movie/show grouped by genre, on the IMDb platform. Movies have usually more interaction from the audience. The most voted genres are action, crime, horror, scifi, and thriller. Western movies have such a high number of votes due to *Django Unchained*'s 1,5 million votes being one of the only 28 movies from this genre.

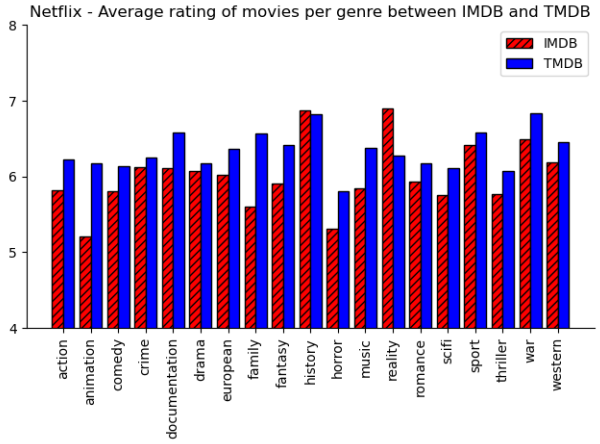

Fig. 10. Average movie score per genre IMDb vs TMDB.

Figure 10 compares the average score of movies, grouped by genre, on the platforms IMDb and TMDB. No genre has a difference higher than 1 point on a 1-10 scale. TMDB has generally better scores for the same kind of movie.

Figure 11 shows a box plot regarding the number of words in the title of movies and TV shows. We can conclude that the middle half is very similar between the two and movies have more extreme outliers than TV shows.

Figure 12 shows a box plot regarding the number of words in the description of movies and TV shows. The middle half of the TV shows has less variance than the movies, although there are many outliers in both cases.

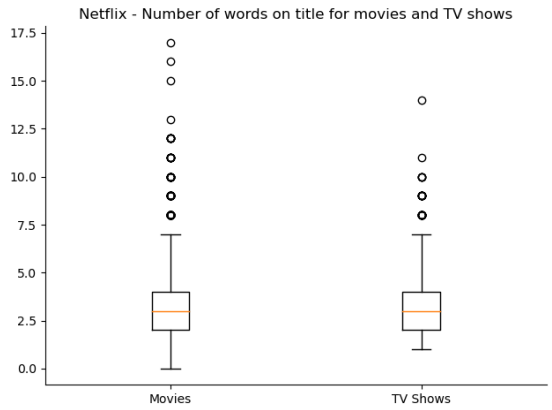

Fig. 11. Number of words in the title.

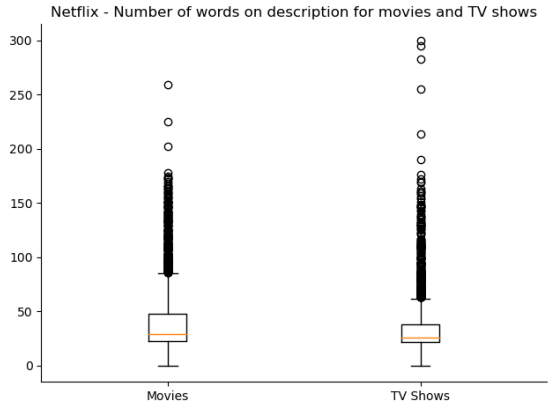

Fig. 12. Number of words in the description.

G. Search scenarios

The next step in this project is to implement and use an information retrieval tool on the dataset. Below are some examples of interesting retrieval scenarios:

- What titles are on Netflix about cars?
- What titles are on Netflix related to Christmas and family?
- In what titles has Christina Hendricks participated since 2016?
- What titles are similar to 'Fast & Furious Spy Racers' by their description?
- What actors are similar to George Clooney by genres of titles he's participated in?
- Who participates in the movie 'Bonnie and Clyde'?

III. Information Retrieval

This section describes the implementation and use of an information retrieval tool, Solr, on the dataset and its exploration with some queries that answer the previously defined information needs. Finally, these queries are evaluated using measures like precision and recall.

A. Collection and Indexing

A document in this system corresponds to a title. In the previous stage, the dataset was divided into 4 entities. However, it is not appropriate for the chosen information retrieval framework, Solr, so we joined all data. Genres and scores were directly merged into titles as they had a one-to-one relationship, but that's not the case with credits. As we know, a title can have multiple credits and vice-versa, but they cannot be simply joined as a list attribute once a credit has multiple fields (person_id, movie_id, name, character, and role). The correct way to join this kind of data is to create nested documents, resulting in the credits being the children of a parent title.

For the indexing process, every attribute has a corresponding field in the schema, and also there's a field for nested documents resulting in a total of 19 fields (table I specifies the fields used in the schema. We decided to index the main search fields: title, description, genres, name and character. For these fields, we built a new field type, to better respond to the search process, called *Text* where we defined:

- **Classic Tokenizer** as this tokenizer behaves as the Standard but doesn't split words with hyphens (which occurs, for example, in Star Wars with R2-D2: we don't want the tokenizer to split it into "R2", "D2");
- **ASCII Folding Filter** to convert characters that are not ASCII;
- **Lower Case Filter** to allow match despite casing;
- **English Minimal Stem Filter** to allow a match between singular and plural forms of words;
- **Classic Filter** to remove possessives;
- **Porter Stem Filter** to apply stemming;
- **Synonym Graph Filter** to match tokens with their synonyms, associating, for example, *family* with *home*, which is used in the second information need in the next section;
- **Beider-Morse Filter** to allow identification of similar names, even if they are spelled differently or in different languages.

B. Retrieval

This section explores queries that answer the system information needs, previously defined at "Search scenarios". Below, the retrieval process is described for each information need, as well as the implemented boosts and types of search, such as phrase match, fuzzy and proximity search. For each information need, a table of results is presented including information about the rank and relevance of the retrieved data.

The first information need corresponds to a basic search scenario where the user wants to find out what titles are there in Netflix about cars, not all but only the good ones. To achieve this, the system performs a query for documents containing the term 'cars' on the *title* and/or *description* and filters the *imdb_score* to 7,0 or higher. For search boosting, documents where 'cars' appear in the *title* will have their score boosted relative to the ones where the term only appears in the *description*. This allows better indexing as the search term could appear in the movie/show *description* (for example, to describe a scene where there is a car accident that impacts the story) even though the movie/show isn't about cars. The configuration and results for this query are presented in tables II and III, respectively. The results are the same for both normal and boosted queries.

Field	Type	Indexed
title	Text	X
description	Text	X
type	String	
release_year	Integer	
age_certification	String	
runtime	Integer	
seasons	Integer	
genres	Text	X
imdb_id	String	
imdb_score	Float	
imdb_votes	Integer	
tmdb_votes	Integer	
tmdb_score	Float	
tmdb_popularity	Float	
person_id	Integer	
name	Text	X
character	Text	X
role	String	
content_type	String	

TABLE I
SCHEMA FIELDS

-	Normal	Boosted
Query string	cars	cars
Configuration	qf: title description	qf: title^3 $description$

TABLE II
QUERY 1 CONFIGURATION

The second information need corresponds to the search of a title using two terms, where the user searches for *Christmas family* related titles. This query is also performed for the *title* and *description* fields. For search boosting, as we are using more than one word and we want to value the proximity of the terms, we included a phrase slop of 4, meaning that documents with 4 or fewer words between *Christmas* and *family* will have

-	Normal		Boosted	
Rank	Title	R	Title	R
1	Fastest Car	X	Fastest Car	X
2	Car Masters: Rust to Riches	X	Car Masters: Rust to Riches	X
3	Comedians in Cars Getting Coffee	X	Comedians in Cars Getting Coffee	X
4	Top Gear	X	Top Gear	X
5	The Lies Within		The Lies Within	
6	Pui Pui Molcar		Pui Pui Molcar	
7	Okko's Inn		Okko's Inn	
8	Dirty Money		Dirty Money	
9	American Vandal		American Vandal	
10	Rust Valley Restorers	X	Rust Valley Restorers	X

TABLE III
QUERY 1 RESULTS

-	Normal	Boosted
Query string	christmas family	christmas family
Configuration3	qf: title description	qf: title description pf: title description ps: 4

TABLE IV
QUERY 2 CONFIGURATION

-	Normal		Boosted	
Rank	Title	R	Title	R
1	A Family Reunion Christmas	X	A Family Reunion Christmas	X
2	The Claus Family	X	Home for Christmas	X
3	Home for Christmas	X	Three Days of Christmas	X
4	Holiday Home Makeover with Mr. Christmas	X	Angela's Christmas	X
5	You Are My Home		A Princess for Christmas	
6	DreamWorks Home: For the Holidays		The Claus Family	X
7	National Lampoon's Christmas Vacation		Holiday Home Makeover with Mr. Christmas	X
8	48 Christmas Wishes		You Are My Home	
9	Father Christmas Is Back	X	DreamWorks Home: For the Holidays	
10	Three Days of Christmas	X	National Lampoon's Christmas Vacation	

TABLE V
QUERY 2 RESULTS

a higher value. The configuration and results for this query are presented in tables IV and V, respectively.

In this third information need, the user wants to find the titles in which Christina Hendricks participated since 2016. In this query, the lucene parser has to be used as we are using nested documents and it is the only way to search by children and obtain the parent document. The lucene parser doesn't accept search boosts. In the first configuration, phrase match is used, as we look for the specific actress. In a second configuration, we introduced fuzzy search for *Christina*, which, not only returned results like *Christina Huertes* and *Christina Hendricks*, but also *Cristina Banegas* and *Kristina Agosti*. The configuration and results for this query are presented in tables VI and VII, respectively.

-	Normal	Fuzzy
Query string	name:"Christina Hendricks"	name:"Christina∼ "
Configuration	fq: release_year: [2016 TO *]	fq: release_year: [2016 TO *]

TABLE VI
QUERY 3 CONFIGURATION

-	Normal		Fuzzy	
Rank	Title	R	Title	R
1	Good Girls	X	War Dogs	X
2	Crooked House	X	Pottersville	X
3	Candy Jar	X	Wild District	X
4	Pottersville	X	Si saben cómo me pongo Pa qué me invitan	X
5			Paradise Lost	X
6			Little Boxes	X
7			The Dreamseller	X
8			Our Lovers	X
9			A Futile and Stupid Gesture	X
10			Valor	X

TABLE VII
QUERY 3 RESULTS

-	Normal	Boosted
Query string	A government agency recruits teen driver Tony Toretto and his thrill-seeking friends to infiltrate a criminal street racing circuit as undercover spies	A government agency recruits teen driver^2 Tony Toretto and his thrill-seeking friends$^2 to$ infiltrate a criminal street racing$^2 circuitas$ undercover spies
Configuration	qf: title description	qf: title description

TABLE VIII
QUERY 4 CONFIGURATION

The fourth information need is related to the main objective of this work: a recommendation system for titles. To achieve this, for now, we're using the *description* field to find similarities between movies/shows as it is the one that better describes the title characteristics. If the user wants to watch titles similar to *Fast Furious Spy Racers*, then the query uses its description to search for related documents. Analyzing the description of this movie, in particular, it's expected that the result is going to be something about cars, racing, government and spies (the first result is, obviously, the movie itself). For search boosting, we used term boosts for the terms *driver*, *friends* and *racing* in the movie's description. It is expected that titles with those terms will have a higher value. The configuration and results for this query are presented in tables VIII and IX, respectively.

For the fifth information need, we wanted to get credits similar to a given actor by the genre of titles they participated in, but, using nested documents, we couldn't find a solution to do so in a simple query.

The sixth, and last, information need is about getting the credits of a given movie, in this case, 'Bonnie and Clyde'. This is the opposite of the third information need, as we search by the parent to obtain the children. We are again using phrase match.

C. Evaluation

This section evaluates the previous queries using some metrics: recall, precision at 10 (P@10), average precision (AP) and mean average precision (MAP). In some queries (1, 2 and 4), we used a subset of documents to calculate recall. This subset is equivalent to the first 20 documents retrieved by the boosted query.

-	Normal		Boosted	
Rank	Title	R	Title	R
1	Fast & Furious Spy Racers	X	Fast & Furious Spy Racers	X
2	The Last Laugh		Race: Bubba Wallace	
3	Guru		Rush	
4	Race: Bubba Wallace		Race	X
5	Turbo FAST		Race 2	X
6	Rush		Dragons: Race to the Edge	
7	Race 2	X	Motu Patlu Aur Khazaane Ki Race	
8	Race	X	Thomas & Friends: All Engines Go - Race for the Sodor Cup	
9	Hajwala 2: Mysterious Mission	X	Hajwala 2: Mysterious Mission	X
10	Dragons: Race to the Edge		Sugar Rush	

TABLE IX
QUERY 4 RESULTS

-	Normal
Query string	title:"Bonnie and Clyde"

TABLE X
QUERY 5 CONFIGURATION

Query 1 returns the exact first 10 documents in both the normal and boosted. 5 of these documents are considered relevant: the movie/show is about **cars**. The other 5 just have the term in their title or description to describe a moment of the movie where something related to a car happened, like a car crash. In this query, we assumed that there are a total of 8 relevant documents in total (8 out of the first 20 retrieved from the boosted query), which are also retrieved in the normal query (3 outside the first 10). The results of the evaluation are presented in table XII.

The second query includes two terms: **Christmas** and **family**. In the first 10 documents of both the normal and boosted queries, 6 are considered relevant. The other 4 just refer the terms in title and/or description but the movie/show context isn't what we're looking for. The boosted query has a better AP because it has relevant documents ranked sixth and seventh while the normal query has in ninth and tenth. In this query, we assumed that there are a total of 11 relevant documents in total (11 out of the first 20 retrieved from the boosted query), which are also retrieved in the normal query (5 outside the first 10). The results of the evaluation are presented in table XIII.

-	Normal		
Rank	Name	R	
1	Warren Beatty	X	
2	Faye Dunaway	X	
3	Guru	X	
4	Michael J. Pollard	X	
5	Gene Hackman	X	
6	Estelle Parsons	X	
7	Denver Pyle	X	
8	Dub Taylor	X	
9	Evans Evans	X	
10	Gene Wilder	X	

TABLE XI
QUERY 5 RESULTS

-	Normal	Boosted
Recall	1	1
P@10	0.50	0.50
AP	0.90	0.90

TABLE XII
QUERY 1 EVALUATION

-	Normal	Boosted
Recall	1	1
P@10	0.60	0.60
AP	0.86	0.95

TABLE XIII
QUERY 2 EVALUATION

Query 3, in its regular form, has only 4 documents retrieved as we're using phrase match and its precision at 4 is 1. The fuzzy query retrieves only relevant documents, resulting in a P@10 of 1 and AP of 1. The results of the evaluation are presented in table XIV.

In the fourth query, we searched for titles similar to 'Fast Furious Spy Racers'. In both normal and boosted queries only 4 documents are considered relevant, while the other ones simply have terms in common in their descriptions. The AP is higher for the boosted query as relevant documents are ranked fourth and fifth, while on the normal query they're ranked seventh and eighth. In this query, we assumed that there are a total of 6 relevant documents in total (6 out of the first 20 retrieved from the boosted query), which are also retrieved in the normal query (2 outside the first 10). The results of the evaluation are presented in table XV.

For the last query, we used phrase match so it was expected that precision would be high and, in fact, it is 1 as all documents retrieved are relevant. The results of the evaluation are presented in table XVI.

Using only metrics from the queries that have normal and boosted versions, the MAP for the normal system is 0.76, while the MAP for the boosted system is 0.83, meaning that

-	Normal	Fuzzy
Recall	1	1
P@10	P@4=1	1
AP	1	1

TABLE XIV
QUERY 3 EVALUATION

-	Normal	Boosted
Recall	1	1
P@10	0.40	0.40
AP	0.53	0.64

TABLE XV
QUERY 4 EVALUATION

-	Normal
Recall	1
P@10	1
AP	1

TABLE XVI
QUERY 5 EVALUATION

the precision is 8% higher in the boosted system.

Figures 13, 14 and 16 show the PR curves obtained for boosted queries 1, 2 and 4, respectively. Figure 15 shows the PR curve for both query 3 (normal and fuzzy) and 5 as their curve is similar.

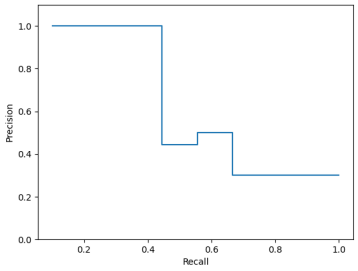

Fig. 13. Precision-recall curve for query 1 boosted.

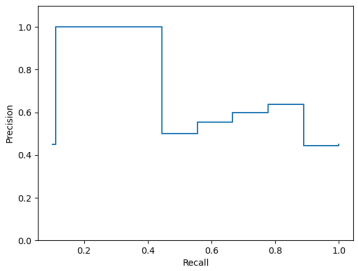

Fig. 14. Precision-recall curve for query 2 boosted.

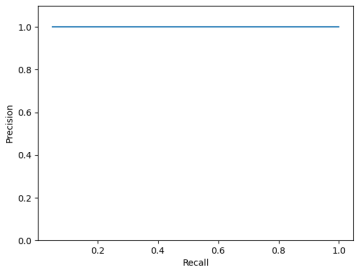

Fig. 15. Precision-recall curve for queries 3 (normal and fuzzy) and 5.

IV. CONCLUSIONS

This document described the processes of data preparation and information retrieval on the dataset. The consequent data operations were described, as well as the defined processing pipeline and makefile. The conceptual data model was presented and the data was analyzed through some histograms and box plots. The indexing of the collection was explained and we were able to respond to 5 of the 6 information needs, even though the precision, in some cases, was not as high as expected, mainly because the terms used for the queries are somewhat general and easy to appear in a movie description, even if the term isn't portrayed in the movie.

The expected work for this stage was successfully concluded. In the next stages, we expect to improve the retrieval process.

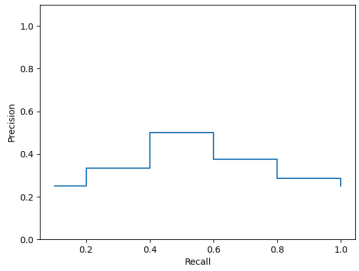

Fig. 16. Precision-recall curve for query 4 boosted.

REFERENCES

[1] –. (2022, Oct) IMDb [Online]. Available: https://www.imdb.com/
[2] –. (2022, Oct) TMDB [Online]. Available: https://www.themoviedb.org/
[3] –. (2022, Oct) WordNet [Online]. Available: https://wordnet.princeton.edu/
[4] –. (2022, Nov) Solr [Online]. Available: https://solr.apache.org/
[5] –. (2022, Nov) Stackoverflow [Online]. Available: https://stackoverflow.com/questions/36587990/how-can-you-retrieve-a-full-nested-document-in-solr
[6] –. (2022, Nov) Sease [Online]. Available: https://sease.io/2019/06/apache-solr-childfilter-transformer.html