RecFlix - TV shows and movies recommendations

1st André Flores
up201907001@fe.up.pt
FEUP

2nd Guilherme Garrido
up201905407@fe.up.pt
FEUP

3rd Luís Lucas
up201904624@fe.up.pt
FEUP

***Abstract*—This work deals with the analysis, processing, characterization, retrieval, and querying of a dataset of TV shows and movies from the main streaming platforms, Netflix, Disney+, HBO Max and Amazon Prime. The goal of this project is to develop an information system that allows users to search for 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. The retrieval tasks were evaluated according to their precision and recall and, finally, a search user interface was developed.**

I. Introduction

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. 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. This paper is split into three parts. In the first, Data Preparation (II), we describe the dataset, data operations, processing pipeline, and conceptual data model and characterize the data using multiple graphs. In the second part, Information Retrieval (III), we present the document definition and the indexing process and analyze the queries defined to answer information needs, evaluating the results using precision and recall metrics. In the last part, Search system improvements (IV), we explain the improvements made to the system, evaluate them and compare them to previous configurations, and, finally, a search user interface was developed using the Vue framework. For the dataset, we chose a Netflix movies/shows one because it has both textual (title, description, genres) and numeric (runtime and scores) data. At a later stage (IV), we complemented the system by adding data from other streaming platforms, such as Disney+, HBO Max and Amazon Prime.

II. Data Preparation

The initial dataset used in this project refers to movies and shows available on Netflix streaming in July 2022, extracted by Victor Soeiro [7]. 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. As an improvement to the search system, we added to our dataset titles and credits from other streaming platforms, such as Disney+, HBO Max and Amazon Prime, resulting in a total of 20394 titles and 294697 credits.

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 (155 records removed), records with no information about age certification (10510 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 1798 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

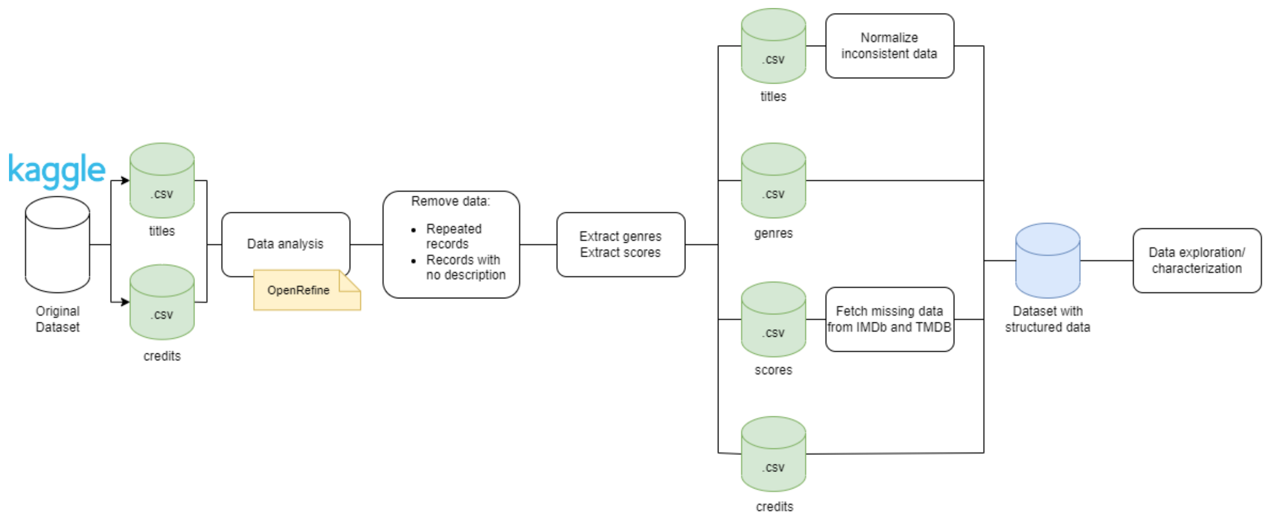


Fig. 1. Final data pipeline summarizing the processing steps.

- 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
- 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

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.

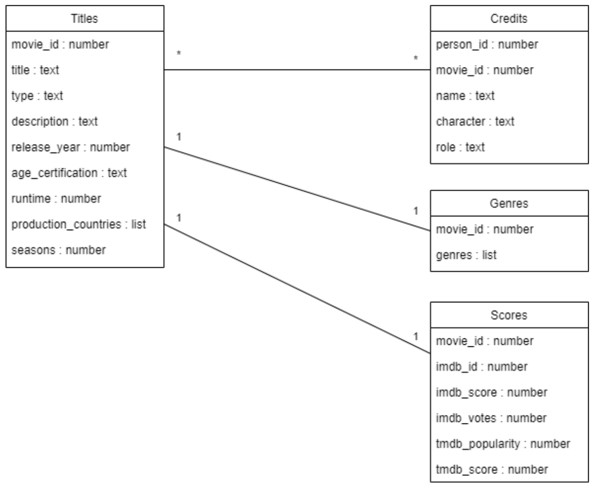


Fig. 2. Conceptual data model.

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 initial dataset (only Netflix data).

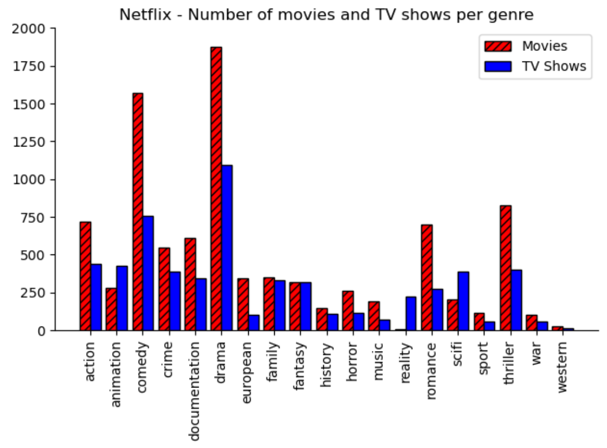


Fig. 3. Movies/shows distribution per genre.

The titles.csv file has almost 6000 registers. Figure 3 shows the distribution by genre 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).

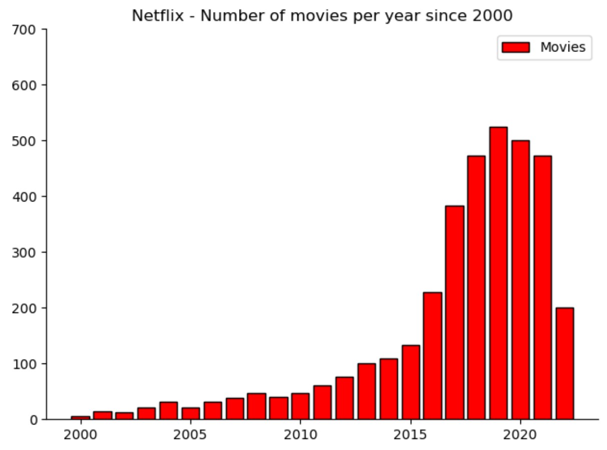


Fig. 4. Movies distribution per year since 2000.

Figures 6 and 7 show, in histograms, the average score, on the IMDb platform, of movies and shows, respectively,

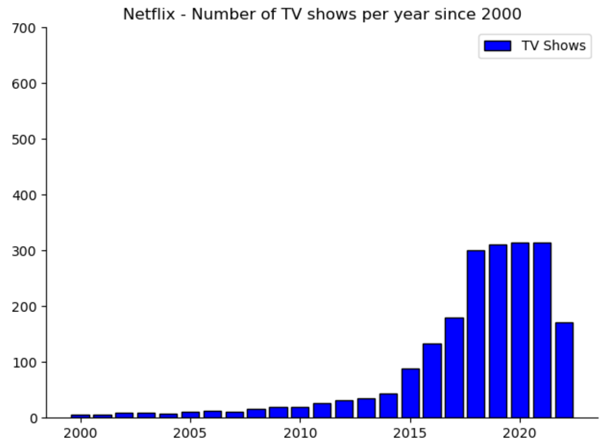


Fig. 5. Shows distribution per year since 2000.

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.

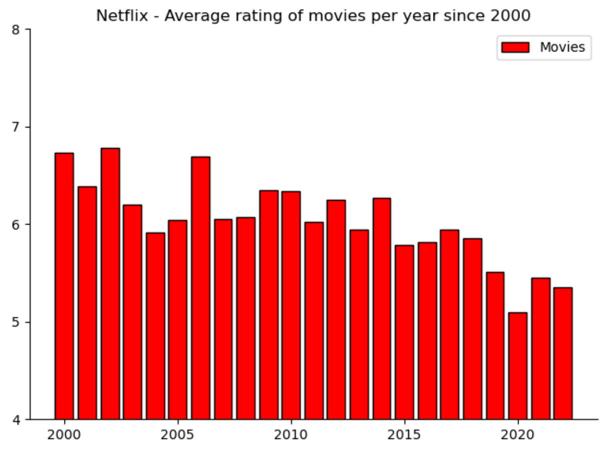


Fig. 6. Average movie score per year since 2000.

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, and thriller have the biggest differences between movies and shows.

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.

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.

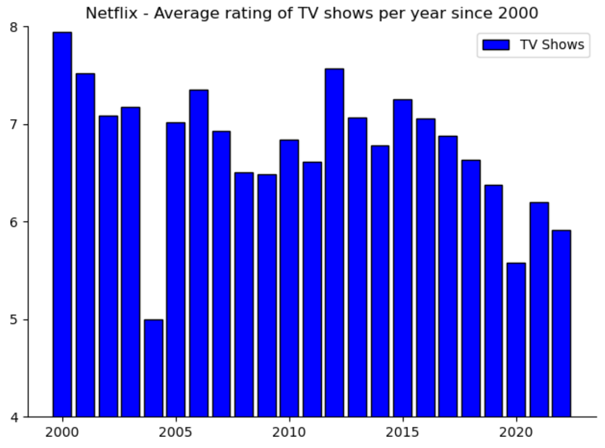


Fig. 7. Average TV show score per year since 2000.

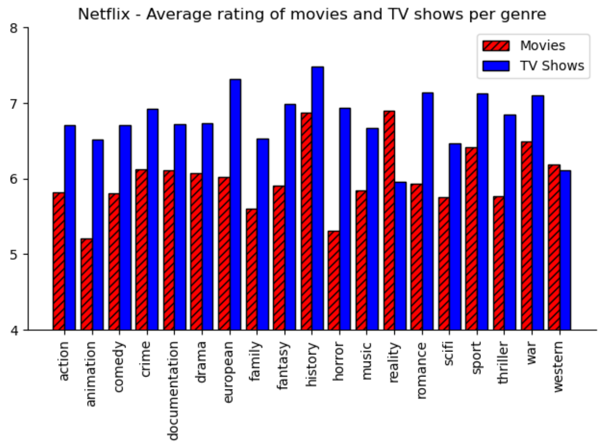


Fig. 8. Average movie/show score per genre.

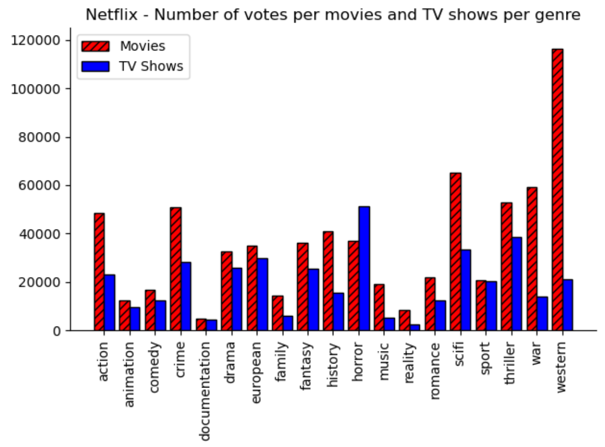


Fig. 9. Average votes on movie/show per genre.

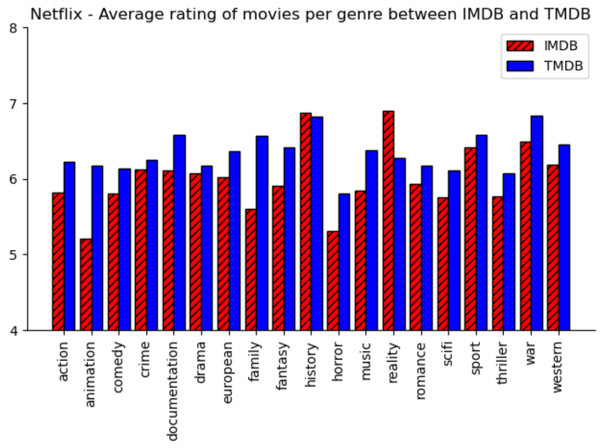


Fig. 10. Average movie score per genre IMDb vs TMDB.

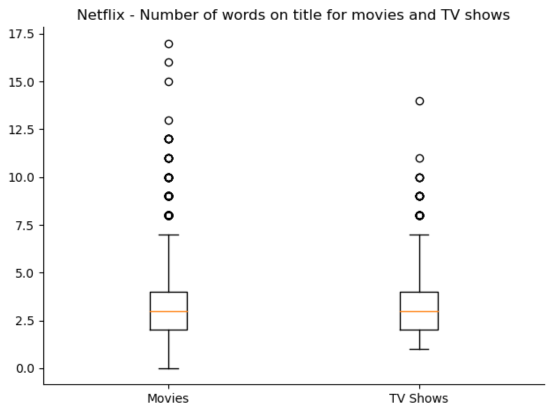


Fig. 11. Number of words in the title.

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.

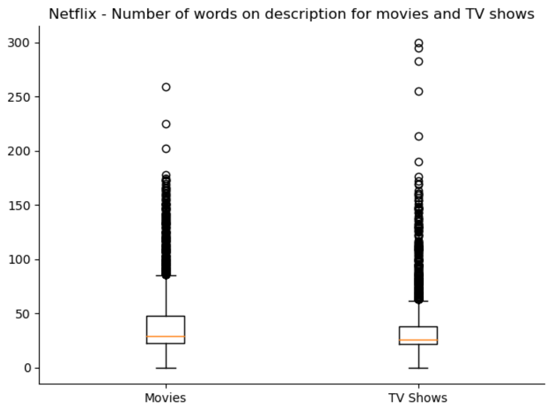


Fig. 12. Number of words in the description.

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.

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 there about cars?
- What titles are there 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?
- Who participates in the movie 'Bonnie and Clyde'?
- What actors are similar to George Clooney by genres of titles he's participated in?

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;

TABLE I
Schema fields

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	

- **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" (II-G). 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 for a schemaless system, a normal system with a schema and a boosted system. These results are relative to the initial dataset, which contains only data from the Netflix platform. The whole dataset is evaluated in the Search System improvements section.

The first information need corresponds to a basic search scenario where the user wants to find out what titles are there on 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 the three systems.

TABLE II
QUERY 'CARS' TITLES PARAMETER CONFIGURATION

-	No schema and Normal	Boosted
Query string	cars	cars
Configuration	qf: title description	qf: title$^3 description$

TABLE III
QUERY 'CARS' TITLES TOP 10 RESULTS

-	No schema	Normal	Boosted
Rank	Relevant	Relevant	Relevant
1	X	X	X
2	X	X	X
3	X	X	X
4	X	X	X
5			
6			
7			
8			
9			
10	X	X	X

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 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 schemaless and normal systems, phrase match is used, as we look for the specific actress. For the boosted system, 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.

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. For search boosting, we used term boosts, increasing the weights of 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.

The fifth information need is about getting the credits of a given movie, in this case, 'Bonnie and Clyde'. This is the

TABLE IV
QUERY 'CHRISTMAS FAMILY' TITLES PARAMETER CONFIGURATION

-	No schema and Normal	Boosted
Query string	christmas family	christmas family
Configuration	qf: title description	qf: title description pf: title description ps: 4

TABLE V
QUERY 'CHRISTMAS FAMILY' TITLES TOP 10 RESULTS

-	No schema	Normal	Boosted
Rank	Relevant	Relevant	Relevant
1	X	X	X
2	X	X	X
3	X	X	X
4	X	X	X
5			
6			X
7			X
8			
9	X	X	
10	X	X	

TABLE VI
QUERY 'CHRISTINA HENDRICKS' TITLES PARAMETER CONFIGURATION

-	No schema and Normal	Fuzzy
Query string	name:"Christina Hendricks"	name:"Christina∼ "
Configuration	fq: release_year: [2016 TO *]	fq: release_year: [2016 TO *]

TABLE VII
QUERY 'CHRISTINA HENDRICKS' TITLES TOP 10 RESULTS

-	No schema	Normal	Fuzzy
Rank	Relevant	Relevant	Relevant
1	X	X	X
2	X	X	X
3	X	X	X
4	X	X	X
5			X
6			X
7			X
8			X
9			X
10			X

TABLE VIII
QUERY 'FAST & FURIOUS' DESCRIPTION PARAMETER CONFIGURATION

-	No schema and 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$^2 circuit$ as undercover spies
Configuration	qf: title description	qf: title description

TABLE IX
QUERY 'FAST & FURIOUS' DESCRIPTION TOP 10 RESULTS

-	No schema	Normal	Boosted
Rank	Relevant	Relevant	Relevant
1	X	X	X
2			
3			
4			X
5			X
6			
7	X	X	
8	X	X	
9	X	X	X
10			

TABLE X
QUERY 'BONNIE AND CLYDE' CREDITS PARAMETER CONFIGURATION

-	No schema and Normal
Query string	title:"Bonnie and Clyde"

opposite of the third information need, as we search by the parent to obtain the children. We are again using phrase match.

For the last 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.

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. We only considered a register as relevant if it would satisfy the user's needs (in our opinion).

Query 1 ('cars' titles) returns the exact same first 10 documents for the 3 system configurations. 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 other systems (3 outside the first 10). The results of the evaluation are presented in table XII.

The second query ('Christmas family' titles) includes two terms: **Christmas** and **family**. In the first 10 retrieved documents in the 3 systems, 6 are considered relevant. The other 4 just refer the terms in the 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 schemaless and normal queries (5 outside the first 10). The results of the evaluation are presented in table XIII.

Query 3 ('Christina Hendricks' titles), in the non-boosted systems, 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 an 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 the three configurations, 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 others 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 schemaless and normal queries (2 outside the first 10). The results of the evaluation are presented in table XV.

For the last query ('Bonnie and Clyde' credits), 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.

TABLE XI
QUERY 'BONNIE AND CLYDE' CREDITS TOP 10 RESULTS

-	No schema	Normal
Rank	Relevant	Relevant
1	X	X
2	X	X
3	X	X
4	X	X
5	X	X
6	X	X
7	X	X
8	X	X
9	X	X
10	X	X

TABLE XII
SYSTEM EVALUATION FOR QUERY 'CARS' TITLES

-	No schema	Normal	Boosted
Recall	1	1	1
P@10	0.50	0.50	0.50
AP	0.90	0.90	0.90

TABLE XIII
SYSTEM EVALUATION FOR QUERY 'CHRISTMAS FAMILY' TITLES

-	No schema	Normal	Boosted
Recall	1	1	1
P@10	0.60	0.60	0.60
AP	0.86	0.86	0.95

TABLE XIV
SYSTEM EVALUATION FOR QUERY 'CHRISTINA HENDRICKS' TITLES

-	No schema	Normal	Fuzzy
Recall	1	1	1
P@10	P@4=1	P@4=1	1
AP	1	1	1

TABLE XV
SYSTEM EVALUATION FOR QUERY 'FAST & FURIOUS' DESCRIPTION

-	No schema	Normal	Boosted
Recall	1	1	1
P@10	0.40	0.40	0.40
AP	0.53	0.53	0.64

TABLE XVI
SYSTEM EVALUATION FOR QUERY 'BONNIE AND CLYDE' CREDITS

-	No schema	Normal
Recall	1	1
P@10	1	1
AP	1	1

Using only metrics from the queries that have an equivalent normal and boosted versions (queries 1, 2 and 4), the MAP for the schemaless and normal systems is 0.76, while the MAP for the boosted system is 0.83, meaning that the precision is 8% higher in the boosted system. The system with no schema may seem to produce the same results as the one with a schema configured, but that's true, in the majority of cases, for the first 10 retrieved documents. For example, in the second query, the *"Grumpy Christmas"* movie, which is ranked 11th in the normal system and 12th in boosted one, doesn't make the top 20 for the schemaless configuration.

Figures 13, 14 and 15 show the Precision-Recall curves obtained for boosted queries 1, 2 and 4, respectively. Figure 16 shows the P-R curve for both query 3 and 5 as their curve is similar in all systems.

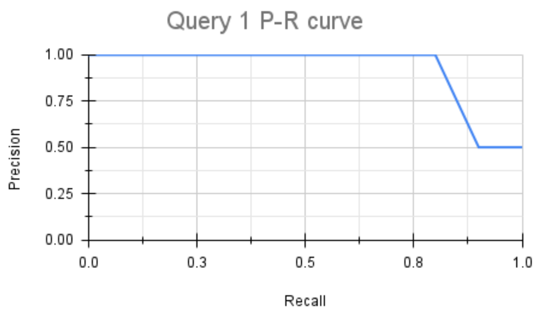


Fig. 13. Precision-recall curve for query 1 in the boosted system.

IV. SEARCH SYSTEM IMPROVEMENTS

To improve our search system we decided to operate in multiple areas. The only point that was left open was the fifth information need. We decided to approach this and also talk about our synonyms file, which is one of the most important elements in our system's performance. Additionally, we added information sources from other streaming platforms and developed a user interface.

A. Query for the last information need

The objective of this query is to find actors that are similar to George Clooney in terms of titles they've been in. To achieve this, first, we need to get the titles George Clooney participated in to then find other actors who have also participated in these titles. This means that we need two separate queries, where the second one uses the output of the first. So, we developed a python script that first retrieves the titles where George Clooney is in the credits and then, retrieves the actors of that titles. Analyzing the results of the query in the 3 systems, all retrieved documents are relevant because all corresponding actors/actresses participated in at least one title with George Clooney. The retrieved documents are the same for both schemaless and normal system configurations. The top 10 results and the evaluation metrics of this query are presented in tables XVII and XVIII, respectively. The Precision-Recall curve for this query is similar to the one for queries 3 and 5

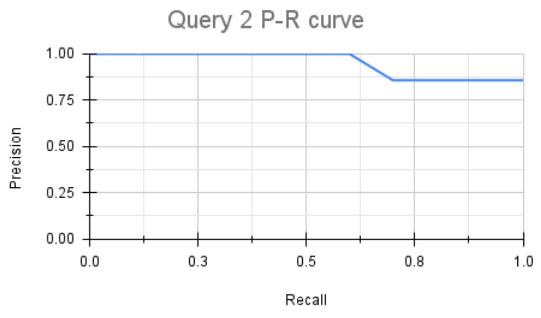


Fig. 14. Precision-recall curve for query 2 in the boosted system.

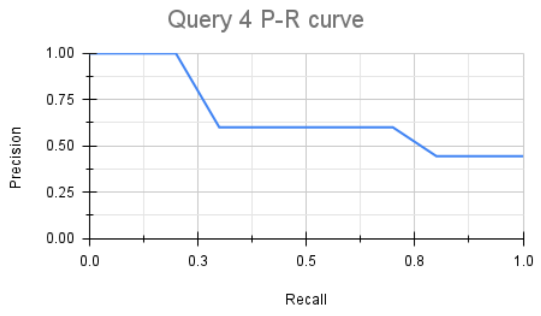


Fig. 15. Precision-recall curve for query 4 in the boosted system.

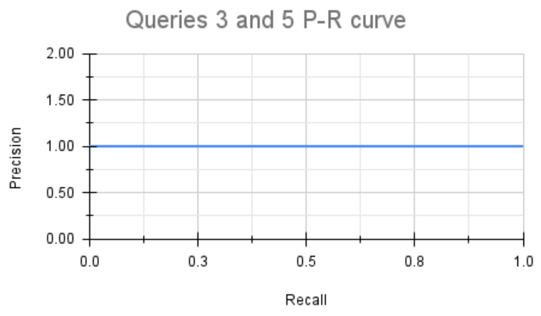


Fig. 16. Precision-recall curve for queries 3 and 5 in all systems.

TABLE XVII
QUERY 'GEORGE CLOONEY' SIMILAR CREDITS TOP 10 RESULTS

-	No schema	Normal
Rank	Relevant	Relevant
1	X	X
2	X	X
3	X	X
4	X	X
5	X	X
6	X	X
7	X	X
8	X	X
9	X	X
10	X	X

TABLE XVIII
SYSTEM EVALUATION FOR QUERY 'GEORGE CLOONEY' SIMILAR CREDITS

-	No schema	Normal
Recall	1	1
P@10	1	1
AP	1	1

(figure 16).

B. Synonyms file

We obtained our synonyms file from WordNet [3] and we had to transform it into a Solr compatible format through a Prolog program.

C. Additional information sources

We added 3 new datasets to our project, which were also made available by Victor Soeiro [7] and have the same structure as the initial one. Our dataset, now, has titles and credits from other streaming platforms, namely Disney+, HBO Max and Amazon Prime, in addition to the original Netflix data. Before indexing the data, we followed the operations described in the pipeline and updated the values in the Data operations section (II-B).

In the first query, where we search for the term 'cars' in the title and description (see query 1 configuration in table II), more data doesn't mean better results, but it meant more results in total, which is also important, increasing the offer to the user. Looking at the first 20 documents retrieved for each of the system variations (table XIX), we can see that, with the added data, the relevance of the higher ranked documents (top 4) is lower (dropping the average precision), but looking outside the top 10 we obtain almost the double of relevant results when compared to the version with the initial dataset, originating a total of 12 (only 11 for the boosted system) relevant documents in first 20. In terms of evaluation (table XX), the precision at 10 doesn't have big improvements but considering precision at 20 we see better results consistently. The recall is the same because the 11 results retrieved by the boosted system are also retrieved by the schemaless and normal systems. Figure 17 shows the Precision-Recall curves for the boosted query in both system variations.

In the second query, where we search for 'Christmas family' related titles (see query 2 configuration in table IV), we

TABLE XIX
QUERY 'CARS' TITLES TOP 10 RESULTS (WITH NEW DATA)

-	Netflix data only			All data		
-	No	Normal	Boosted	No	Normal	Boosted
Rank	Relevant	Relevant	Relevant	Relevant	Relevant	Relevant
1	X	X	X	X		
2	X	X	X	X		
3	X	X	X			X
4	X	X	X	X	X	
5				X	X	X
6					X	X
7				X	X	
8				X	X	
9				X		X
10	X	X	X			
TOP 20	7	7	7	12	12	11

TABLE XX
SYSTEM EVALUATION FOR QUERY 'CARS' TITLES (WITH NEW DATA)

-	Netflix data only			All data		
-	No	Normal	Boosted	No	Normal	Boosted
Recall	1	1	1	1	1	1
P@10	0.50	0.50	0.50	0.70	0.50	0.40
P@20	0.40	0.40	0.40	0.60	0.60	0.55
AP (@10)	0.90	0.90	0.90	0.83	0.47	0.42

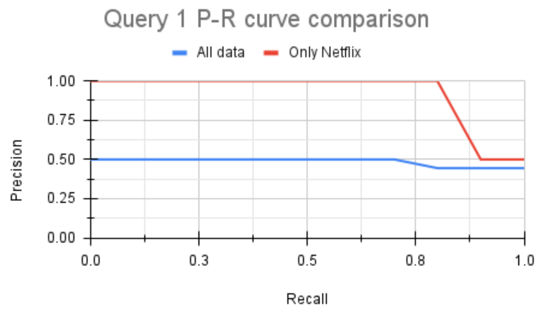


Fig. 17. Precision-Recall curves for query 1 boosted in both system variations

TABLE XXI
QUERY 'CHRISTMAS FAMILY' TITLES TOP 10 RESULTS (WITH NEW DATA)

-	Netflix data only			All data		
-	No	Normal	Boosted	No	Normal	Boosted
Rank	Relevant	Relevant	Relevant	Relevant	Relevant	Relevant
1	X	X	X		X	
2	X	X	X	X	X	
3	X	X	X	X	X	X
4	X	X	X	X		X
5						X
6			X	X		
7			X	X	X	X
8				X		X
9	X	X			X	X
10	X	X		X	X	
TOP 20	10	11	11	12	12	11

TABLE XXII
System evaluation for query 'Christmas family' titles (with new data)

-	Netflix data only			All data		
-	No	Normal	Boosted	No	Normal	Boosted
Recall	1	1	1	1	1	1
P@10	0.60	0.60	0.60	0.70	0.60	0.60
P@20	0.50	0.55	0.55	0.60	0.60	0.55
AP (@10)	0.86	0.86	0.95	0.68	0.79	0.55

expected to obtain a better precision using the proximity search with more data but it also retrieved 11 relevant documents in the first 20 (table XXI). In addition, also the ranking of these is worse, dropping the average precision from 0.95 to 0.55. For example, we didn't consider relevant the title where the description is "... he tells you all he's learned about Christmas, Halloween, Family Day, Valentine's Day ..." because we believe that that isn't what the user wants to find when he searches for 'Christmas family' related titles. Without the boost, this query obtains more relevant results in the top 20 with more data, increasing the precision at 20 (table XXII). The recall keeps the maximum values as the relevant documents retrieved by the boosted query in the top 20 are also retrieved by the other system configurations. Figure 18 shows the P-R curves for the boosted query in both system variations.

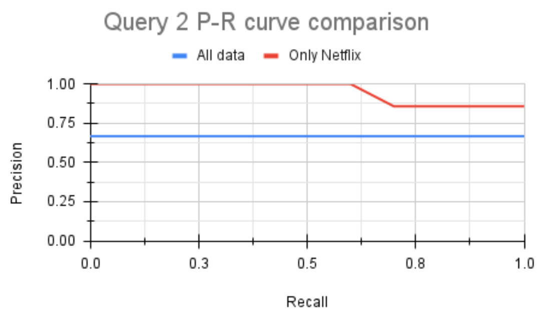


Fig. 18. Precision-recall curve for query 2 boosted in both system variations

In the third query, where the user searches for titles where *Christina Hendricks* participated, the schemaless and normal systems retrieved a total of 8 documents, while the previous version only retrieved 4. This was expected because more titles lead to a higher probability of an actor having more appearances. The fuzzy search query for the term *Christina* (see query 3 configuration, table VI) retrieved only relevant documents in the first 20 retrieved, which we were expecting because the previous version also did. In terms of evaluation, this improved version obtains the maximum values in every metric and its P-R curve is the same as the previous version (figure 16).

In the fourth query, where we search for titles similar to 'Fast & Furious Spy Racers' by their description, we were hoping to find the other movies of the 'Fast & Furious' saga (available now in the dataset) well ranked in the retrieved documents, but that was not the case. Looking at the description of these movies, we observe that, in fact, they don't have much in common and the boosted terms also rarely appear in the other movies' descriptions. For example, the movie 'The Fast and the Furious: Tokyo Drift' has the following description: *In order to avoid a jail sentence, Sean Boswell heads to Tokyo to live with his military father. In a low-rent section of the city, Shaun gets caught up in the underground world of drift racing*. Comparing it to the description of the movie that we are using for the search, we can only find a matching term, *racing*, and, thus, we can't expect that it will be well ranked as many other movies have, at least, one term in common.

TABLE XXIII
Query 'Fast & Furious' description top 10 results (with new data)

-	Netflix data only			All data		
-	No	Normal	Boosted	No	Normal	Boosted
Rank	Relevant	Relevant	Relevant	Relevant	Relevant	Relevant
1	X	X	X	X	X	X
2						
3						
4			X			
5			X		X	X
6				X		
7	X	X			X	
8	X	X		X		
9	X	X	X	X		X
10						
TOP 20	6	6	6	6	4	4

TABLE XXIV
System evaluation for query 'Fast & Furious' description (with new data)

-	Netflix data only			All data		
-	No	Normal	Boosted	No	Normal	Boosted
Recall	1	1	1	1	1	1
P@10	0.40	0.40	0.40	0.40	0.30	0.30
P@20	0.30	0.30	0.30	0.30	0.20	0.20
AP (@10)	0.53	0.53	0.64	0.54	0.61	0.58

Analyzing the relevance of results in table XXIII and the values of precision and recall (table XXIV), we can see that there is no big difference between the version with only Netflix data and the version with all data, other than the fact that this last version only retrieves 4 relevant documents in the top 20 in the normal and boosted systems, while all the other combinations retrieve 6, decreasing the precision at 20. The P-R curves for the boosted query in both system variations are shown in figure 19.

In the fifth query, where the user wants to know the credits for the movie *Bonnie and Clyde* (see query 5 configuration in table X), we expected the improved version to retrieve the exact same results as the system with only Netflix data because there is only one movie with this name in the complete dataset (which is the one from Netflix) and, so, no more credits should be related to it. This is, in fact, the case as both system configurations, schemaless and normal, retrieve the expected 17 documents, corresponding to the 17 credits of the movie,

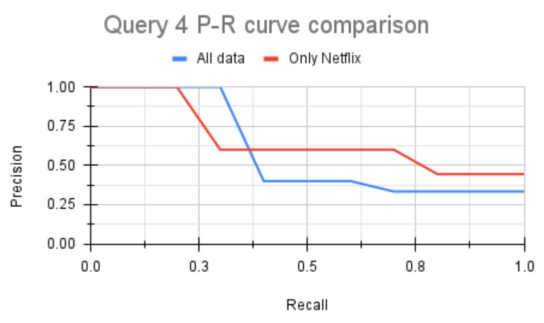


Fig. 19. Precision-recall curve for query 4 boosted in both system variations

that were already retrieved by the previous system. Thus, both precision and recall have not changed (see table XVI), and so, its P-R curve is the same as the one in figure 16.

In the last query, which aims to get credits similar to George Clooney, we were also expecting that this version with more data would keep the precision at 1, and that's what actually happened as all the top 20 documents retrieved (in both schemaless and normal configurations) correspond to credits who participated in a title with George Clooney. The values of the evaluation metrics are the same as the previous version (table XVIII) and therefore the Precision-Recall curve is also the same (figure 16).

Considering the average precision for the first 10 retrieved documents, with the former dataset, the MAP for the schemaless and normal systems is 0.76, while the MAP for the boosted system is 0.83, using queries that have an equivalent normal and boosted versions (queries 1, 2 and 4). With the latest dataset, the MAP for the schemaless system is 0.68, for the normal is 0.62 and for the boosted system is 0.52. In all three systems, the mean average precision dropped, meaning that the rank of relevant results in the top 10 is better using the old dataset.

D. Search User Interface

We focused on making our interface approachable to the average user. It is a single-page website with a search box and selectors for filters that may be applied to the search (figure 20). The search box and the current query are visible to the user at all times. A user may filter their search by selecting whether they want to search by the name or description of a title (figure 21) or by the name of a participant in that title (figure 22), and may also select a range for the IMDb score and release year of the titles they are searching for.

After the user has made their search, the results are presented and may be sorted according to various criteria. For each title, the user may also see the participants of that title (figure 22). If a user clicks on the name of the title they are taken to its IMDb page, if they click on the name of a participant they are taken to its JustWatch page.

The website was developed using the Vue framework [8] and coded in Typescript. We chose this because we were already familiar with this technology stack and it allowed us

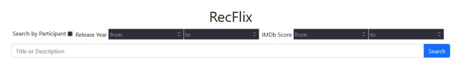
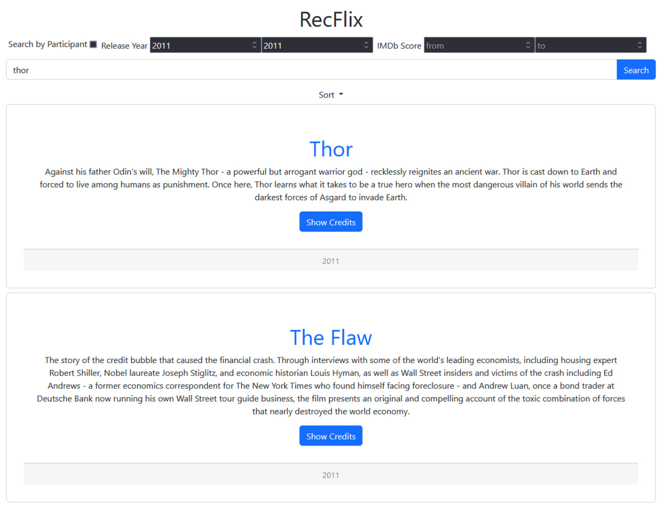


Fig. 20. Landing page for the user interface



Release Year 2011 – 2011

thor

Thor

Against his father Odin's will, The Mighty Thor - a powerful but arrogant warrior god - recklessly reignites an ancient war. Thor is cast down to Earth and forced to live among humans as punishment. Once here, Thor learns what it takes to be a true hero when the most dangerous villain of his world sends the darkest forces of Asgard to invade Earth.

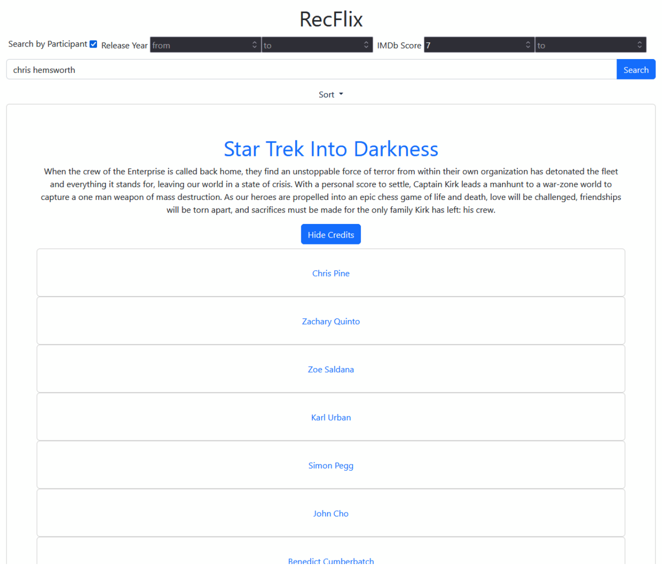
2011

The Flaw

The story of the credit bubble that caused the financial crash. Through interviews with some of the world's leading economists, including housing expert Robert Shiller, Nobel laureate Joseph Stiglitz, and economic historian Louis Hyman, as well as Wall Street insiders and victims of the crash including Ed Andrews - a former economics correspondent for The New York Times who found himself facing foreclosure - and Andrew Luan, once a bond trader at Deutsche Bank now running his own Wall Street tour guide business, the film presents an original and compelling account of the toxic combination of forces that nearly destroyed the world economy.

2011

Fig. 21. Results for a search by title, specifying the release year range



Search by Participant ✓

IMDb Score 7

chris hemsworth

Star Trek Into Darkness

When the crew of the Enterprise is called back home, they find an unstoppable force of terror from within their own organization has detonated the fleet and everything it stands for, leaving our world in a state of crisis. With a personal score to settle, Captain Kirk leads a manhunt to a war-zone world to capture a one man weapon of mass destruction. As our heroes are propelled into an epic chess game of life and death, love will be challenged, friendships will be torn apart, and sacrifices must be made for the only family Kirk has left: his crew.

Chris Pine

Zachary Quinto

Zoe Saldana

Karl Urban

Simon Pegg

John Cho

Benedict Cumberbatch

Fig. 22. Results for a search by a credit, specifying the IMDb score range

TABLE XXV
SEARCH USER INTERFACE QUERY PARAMETERS FOR TITLES

Field	Boost
title	3
description	1
genres	1

TABLE XXVI
SEARCH USER INTERFACE QUERY PARAMETERS FOR CREDITS

Field	Boost
name	1

to easily create a dynamic web page. To help us implement a good user interface we also used Bootstrap for styling the website. The user's inputs are transformed into queries that are sent to Solr using axios. Since the frontend is hosted in a different origin than Solr, this required modifying the *web.xml* file in the Solr instance to allow cross-origin requests.

The queries sent to Solr to search for titles used the parameter configuration shown in table XXV, while the ones to search for credits used the configuration present in table XXVI. We opted for a simple configuration for titles, boosting the title of the movie/show because it was similar to the one used in query 1 (see the first query configuration in table II), which had the best improvement in terms of results with the new dataset. The configuration of the query of credits can only use one field as we are using the lucene parser and nested documents, and so, we think that users would prefer to search for the name of the actor/actress than for the character he/she plays.

V. 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 all 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. We added data from other streaming platforms and compared it with the previous version. The results didn't improve for a subset of the first 20 documents retrieved by each query in Solr. We also developed a search user interface using the Vue framework [8]. To improve the results of our system we could explore some machine learning techniques to help us order the documents and obtain more relevant ones in the higher ranks. We conclude that the work was completed successfully as we addressed all of the initially proposed points.

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

REFERENCES

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