IMDB/Rotten Tomatoes Movies Ratings and Reviews

Caio Nogueira
up201806218@edu.fe.up.pt
University of Porto

Carlos Lousada
up201806302@edu.fe.up.pt
University of Porto

Tomás Mendes
up201806522@edu.fe.up.pt
University of Porto

ABSTRACT

With the constant growth of information available online, it has become extremely important to come up with information systems able to process, index, and search that information effectively. In this article, we will study the case of movies and respective reviews with information available in the 2 most popular movie-related websites on the whole Internet: **Rotten Tomatoes** and **IMDB**. The datasets used are public and available on **Kaggle**. The data was then analyzed and prepared for the retrieval phase. To implement the **information retrieval system**, we used the ***Solr*** platform. After indexing the documents and defining our collection, we chose a set of information needs that would allow us to explore the engine's functionalities and evaluate the search system - what fields are most important to improve the search results.

CCS CONCEPTS

• **Information systems → Evaluation of retrieval results**.

KEYWORDS

Datasets, Data Preparation, Movies, Search Systems, Information Retrieval

1 INTRODUCTION

Cinema was not invented by one person. The first to present projected moving pictures were the Lumière brothers in December 1895 in Paris. They decided to use a device made by themselves, the Cinématographe, which was composed of a camera, a projector and a film printer. Since then, movies evolved tremendously: from basic frames in black and white colours with no sound to an amazing explosion of colours with beautiful soundtracks and dialogues, passing through a golden age (during the 1930s and 1940s) and a rapid improvement of the digital technology (3D, IMAX, CGI, ATMOS, Real|Laser, among other tools).

Bearing this in mind, this article describes the first and second development phases of a movie platform that aims to break any geographical barrier and intends to connect fans of the seventh art from all over the world, using a friendly and intuitive search engine with several applicable filters.

Firstly, it is crucial to collect datasets with relevant information (*Data Collection*). Then, the data should go through a preparation stage (*Data Preparation*), i.e., a variety of refinement tasks, such as data scaling, normalising and cleaning, to be easier to handle. Furthermore, the *Conceptual Model* demonstrates how the dataset is structured and by using and interpreting diverse graphs (*Data Characterization*), we can have a better perception of the collected data.

The second phase is focused on *Information Retrieval*. Initially, we compared the most relevant search engines - *Solr* and *Elasticsearch*. This process is described as *Information Retrieval Tool Selection*. Then, the collections and documents are specified in *Collections and Documents*, describing our different approaches to solve our problem of joining movies and their respective reviews. In the following sections, the field types were applied to the respective relevant fields (*Indexing Process*) and, finally, to evaluate the system's performance, five different information needs were defined (*Retrieval Process*) and their results were discussed based on the configurations set.

2 DATA COLLECTION

The data used for the information retrieval system was collected from *Kaggle* [15], which is a well-known data science website that allows users to publish and use free datasets in a collaborative environment.

In order to collect the necessary data, we downloaded two different datasets from *Kaggle*: one of them containing information scrapped from the IMDB's database[3] and the other from the Rotten Tomatoes's database[4].

The IMDB dataset contains information about over 85000 movies, dated between 1915 and 2020 (when both datasets were scrapped). The data is composed of 4 different *.csv* files, from which we selected the ratings and the movies' general information: title, actors, directors, release date, age suitability. By doing this, we can retrieve information about the ratings given to the movies.

The Rotten Tomatoes dataset is less extensive when it comes to the number of movies: it contains information about roughly 17,000 movies. Despite covering fewer movies, we can use this dataset to collect textual critics for each movie, which will be useful. In addition, we also collected numeric classification for each movie, such as the *Tomatometer* score and *Audience Score* (rating given by the users of Rotten Tomatoes). For these reasons, we can say that the datasets complement each other.

Both datasets are owned by the same *Kaggle* user: Stefano Leone (a data Analyst at a financial company). The fact that the data was collected by the same developer using the same methods (Python with the Requests [10] library) proved to be a huge advantage as the datasets have a very high usability score and the data types are consistent.

3 DATA PREPARATION

After having decided which datasets would be used for the information retrieval system, we needed to refine the data. This section consists of the different steps taken to prepare the data for the information retrieval phase.

For this task, we used Pandas [9] library from Python, which allows reading, writing and manipulation of *csv* files.

3.1 Merging

First of all, since we selected two independent datasets we would need to merge them into one. For this reason, we could only use

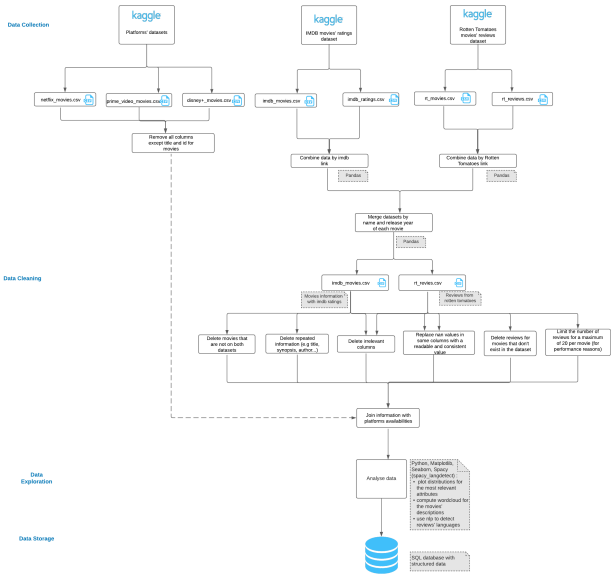

Figure 1: Pipeline design

rows about movies that existed in both datasets. However, before merging, we dropped the common attributes (movie title, director, actors, genre, duration).

The next step in this process was the proper merging of the datasets. We decided to join the files by title and release date.

The merging process resulted in a *csv* file containing approximately 9,000 movies and over 80 columns. In the reviews file, we dropped the rows that didn't refer to movies that existed in the merged dataset.

3.2 Refinement

After having merged the data, we started looking into the different columns. Since some of the columns had NaN (Not a Number) values, we decided to replace them with a more readable text ("Unknown" or "Not defined") depending on the respective attribute. At this point, we had over 80 columns, which translates into an excessive amount of data (per movie) that would not be used further on, so we disposed of it: some columns had specific data that we were not planning to keep for the information retrieval phase (e.g average score among females between 18 and 30 years old), or even redundant data: for example, the attribute *certified fresh* is true for movies whose *Tomatometer score* is above 85

The critics dataset, on the other hand, contained a reasonable amount of columns (8), but an excessive amount of reviews per movie (about 500,000 reviews in total). For performance reasons, we limited the maximum amount of critics for each movie to 20. In order to get the most relevant ones, we sorted the reviews by top critic and text length. This means that if a movie has, for instance, 50 reviews, where **1)** 15 of them are top critics and the rest of them are not, we will pick the 15 top critics and the 5 normal critics with higher word count; **2)** all of them are top critics, we will pick the 20 top critics with higher word count. After this operation, the reviews file was slightly less than 13,0000 rows.

3.3 Enrichment

In order to provide more information and, therefore more possibilities to query the dataset on future milestones, we used another dataset from *Kaggle*, with information about movies from three popular streaming services: *Netflix* [5], *Amazon Prime Video* [6] and *Disney Plus* [7]. The analysis of this new information resulted in three more columns added to the previous (movies) dataset. Each of the additional columns contains boolean values indicating whether the corresponding movie is available on the respective streaming service or not.

4 CONCEPTUAL MODEL

Having completed all the necessary data preparation tasks, we designed a domain conceptual model, which is displayed in Figure 2.

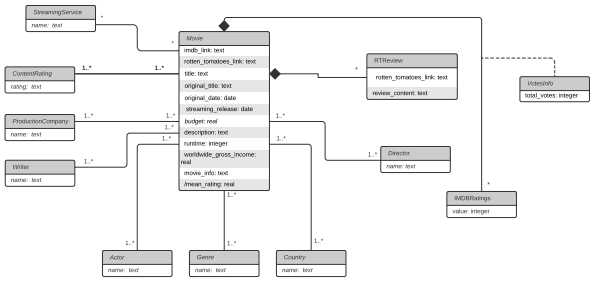

Figure 2: Conceptual Model

Movie is the main class since it connects all other entities in the domain. This class contains relevant information about each movie. The classes *ContentRating*, *ProductionCompany*, *Writer*, *Director*, *Actor*, *Genre* and *Country* are association classes that provide additional information about the movies.

Furthermore, the classes *RTReview* and *IMDBRatings* provide information about the Rotten Tomatoes critics and IMDB ratings, respectively: *RTReviews* entries have as attributes not only the review itself but also some information about the critic who published it (name and reputation on the website). *IMDBRatings'* entries have the rating itself associated (1-10). This table will then be analyzed to compute the mean rating from *IMDb*.

5 SEARCH TASKS

The data collected and prepared can be obtained by the information retrieval system in several ways. Some of the possible retrieval tasks are listed below:

- Search movies about specific historical events
- Search movies that approach certain topics (e.g social issues)
- Search movies based on real events

- Search movies that other users found emotive
- Search for criminal investigation movies
- Search reviews for a movie made by a verified Rotten Tomatoes user (top critic)

6 DATA CHARACTERIZATION

With the data prepared, we started to analyse its distribution and characterization. To complete this task, we used the following Python modules: Matplotlib [2], Seaborn [14] and WordCloud [8].

One of our concerns was to have a wide variety of data. This means that we wanted our final dataset to contain a decent number of movies (and respective reviews and ratings) from a relevant range of years. This proved to be the case as seen in figure 3.

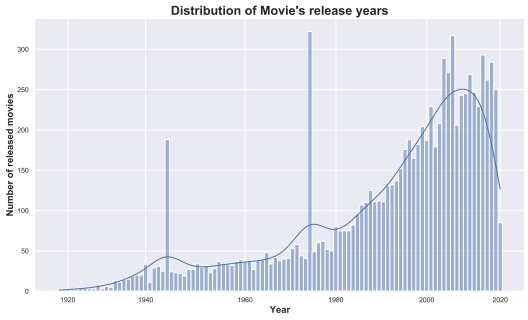

Figure 3: Number of movies per year

As expected, on average, our dataset contains more movies that were released in the 2000s. However, there's still a good representation of movies from the previous century.

Following this line of thought, we also strived for a wide representation of movie production countries, genres and content ratings. This can be observed in figures 4,5 and 6.

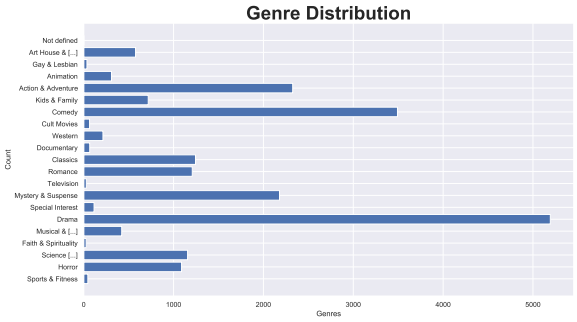

Figure 4: Genre distribution

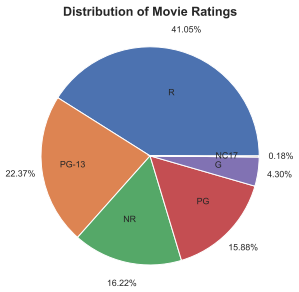

Figure 5: Content-rating distribution

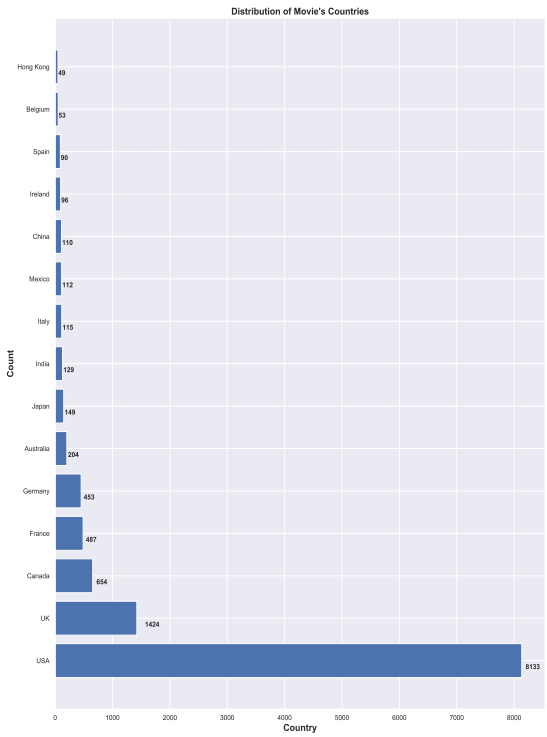

Figure 6: Production Country distribution

Analysing these graphs, we can see that there is a decent distribution of movies in all of the mentioned parameters. We found the results to be very accurate, as it shows Drama, Comedy and Action/Adventure as the most common genres and that R-Rated is the most popular content rating.

Furthermore, regarding rating data, we felt the urge to verify if there's a wide range of ratings and differently rated movies. To do that, we can analyse figures 7,8 and 9.

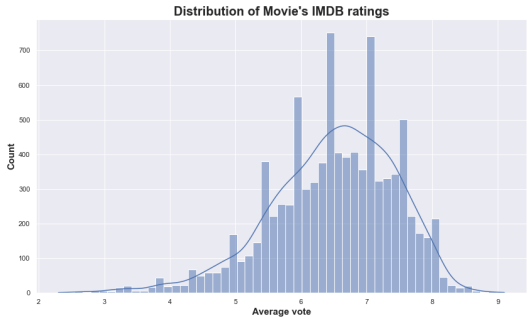

Figure 7: IMDB's rating distribution

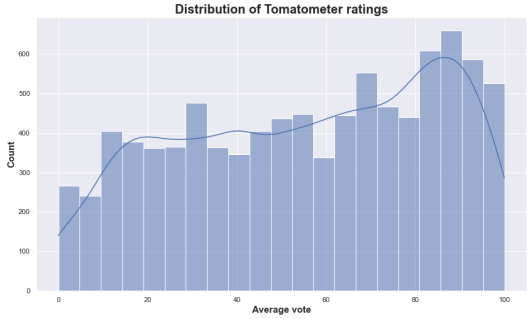

Figure 9: Tomatometer rating distribution

Once again, we can conclude that our dataset includes a wide variety of movies, rated in very different ways in several different ratings.

Finally, we decided to analyse a bar plot to check the evolution of ratings throughout the years, and a Word Cloud plot, to verify that there are several different plot points in the movies taken into account (for instance, family, love, friend, life, etc). Those plots can be seen in figures 10 and 11, respectively.

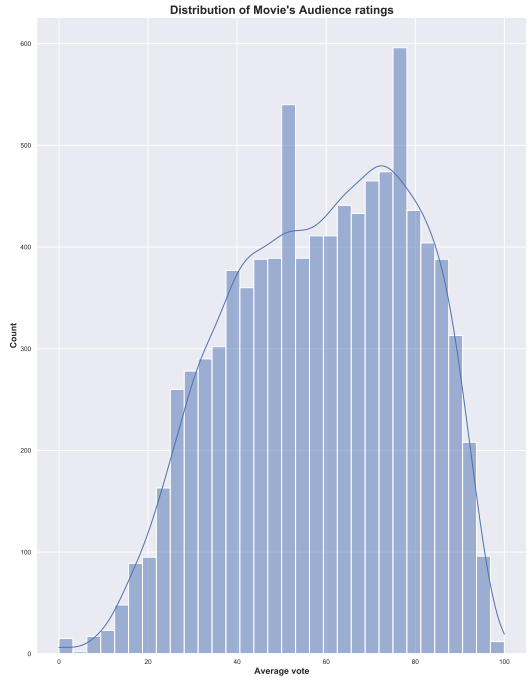

Figure 8: Rotten Tomatoes's Audience rating distribution

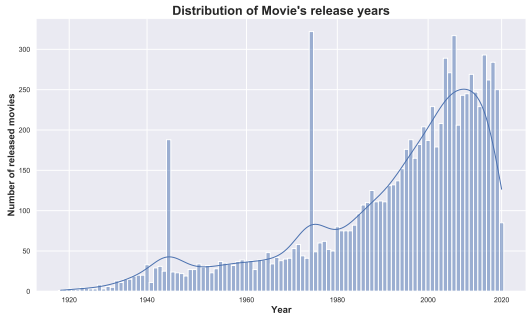

Figure 10: Average IMDB's ratings per year

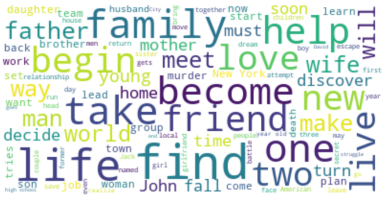

Figure 11: Movie description's Word Cloud

6.1 Text characterization

In our dataset, there are two main textual values for each movie: its description and the critics. Our goal at this point is to improve the understanding of both and, to achieve that objective, we computed the word cloud for the movies' descriptions, which is displayed on **Figure 11**. As to critics, we used *NLP (Natural Language Processing)* with *Spacy* [12] to detect the percentage of English reviews in the dataset. The NLP algorithm pointed out that approximately 99 percent of the reviews were English. The other reviews are written in Portuguese.

7 INFORMATION RETRIEVAL

The main goal of an Information Retrieval System is to retrieve documents from a collection. These documents obtained should be as relevant as possible to a given information need. This work focuses on *ad hoc* retrieval tasks, where the information need is obtained by a specific query.

In order to study and evaluate this process, we will be using 3 different systems:

- A baseline (*schemaless*) system, with basic matching on key fields;
- A system where the main text fields are indexed with the use of filters (*schema*);
- A system with indexed fields and which uses weights to choose the most relevant documents in the retrieval phase (*boosted*).

To evaluate the queries, we will use *Precision at 10* (P@10) which calculates precision at a specific cut point (in this case, 10 documents), and *Average Precision* (AP) that is defined as the average of the precision values obtained for the set of top N documents existing each time a new relevant document is retrieved. To compare the different systems, we will be using *Mean Average Precision* (MAP).

8 INFORMATION RETRIEVAL TOOL SELECTION

In order to develop an Information Retrieval System, we could have opted for 2 different platforms: *Solr* [11] and *ElasticSeach* [1]. Both tools are similar built on *Apache Lucene* [13] and offer a wide variety of features: e.g full-text search, support for No-SQL data and high availability. In this project, we used *Solr* as the search tool, due to the fact that *Solr* supports text search while Elasticsearch is mainly used for analytical querying, filtering, and grouping. For our specific use case, *Solr* is the most adequate tool. The tutorials about *Solr* provided in the course's contents also made it easier to get started with the platform.

9 COLLECTIONS AND DOCUMENTS

The most important documents for this dataset are the movies and their respective reviews.

In order to use multiple types of documents, we followed different approaches. Firstly, we tried to use nested documents: each movie would have as its children, the respective reviews. However, to query this collection we would need to use a complex syntax, that was hard to find on *Solr*'s documentation. In the second attempt, we tried to use different documents without nesting, but we encountered problems when trying to use weights to improve search results.

Our final solution was to have the movies as a single type of document, where each movie has a list of its textual reviews.

10 INDEXING PROCESS

One of the most important steps involved in the development of an Information Retrieval System is the Indexing Process. Before this phase, we needed to decide which fields should be indexed. After analyzing the existing fields, we decided to index the ones which we believed to be useful when searching for a movie.

The indexed and stored fields with numeric values were defined with *Solr*'s default field type *pint* and the dates were defined with the *pdate* field type. The most important textual fields were subjected to an analyzer pipeline, including a *tokenizer* and optional filters to process each token (e.g evaluate them in lower case and without accent marks).

With this in mind, two custom field types were created: *standart_text* and *daterange*.

- The *standart_text* field type is used in the main textual fields. To improve results in the Information Retrieval phase, we applied three different filters: *LowerCaseFilterFactory* to match tokens given by a user query in a non-case-sensitive way; *ASCIIFoldingFilterFactory*, which converts alphabetic, numeric, and symbolic Unicode characters which are not in the Basic Latin Unicode block to their ASCII equivalents, if one exists and *SynonymGraphFilterFactory*, that allows matching a token with its synonyms (defined in a *synonyms.txt* file).
- The *daterange* field type applies the *DateRangeField* that allows indexing date ranges.

The summary of the custom field types can be found in table 1.

Table 1: Custom Field Types

Field Type	Filter	Index	Query
standart_text	ASCIIFoldingFilterFactory	X	X
	LowerCaseFilterFactory	X	X
	SynonymGraphFilterFactory	X	X
daterange	DateRangeField		

Table 2 contains information about the fields relative to each document in the collection.

Table 2: Schema Fields

Field	Field Type	Indexed
imdb_title_id	standard_text	false
rotten_tomatoes_link	standard_text	false
original_title	standard_text	true
original_release_date	daterange	true
streaming_release_date	daterange	true
country	standard_text	true
language	standard_text	true
production_company	standard_text	true
directors	standard_text	true
writer	standard_text	true
budget	standard_text	false
worlwide_gross_income	standard_text	false
genres	standard_text	true
content_rating	standard_text	true
runtime	pint	true
movie_info	standard_text	true
audience_rating	pint	true
tomatometer_rating	pint	true
total_votes	pint	true
mean_vote_imdb	pfloat	true
votes_10	pint	false
votes_9	pint	false
votes_8	pint	false
votes_7	pint	false
votes_6	pint	false
votes_5	pint	false
votes_4	pint	false
votes_3	pint	false
votes_2	pint	false
votes_1	pint	false
available_netflix	standard_text	true
available_prime_video	standard_text	true
available_disney_	standard_text	true
review_content	standard_text	true

11 RETRIEVAL PROCESS

At this point, after finishing the indexing process, we have a complete schema to add the fields with the correct field types and use the *post* tool to populate the collection. To evaluate the system's performance, we defined 5 different information needs, based on the search tasks previously discussed (Section 5). *Solr* offers different query parsers. During the development of this project, we explored the *DisMax* (*Maximal Disjunction*) and EDisMax (*Extended Maximal Disjunction*).

The query fields used were the indexed ones. Moreover, for each information need, we created a list with the relevant documents, thus allowing us to evaluate the results using the metrics referred in section 7.

A. Space movies on Netflix (IN1)

Information need: Movies set in a spatial environment and are available to watch on Netflix.

User story: "As a Netflix client and interested in science fiction movies featuring intergalactic traveling, I want to find movies related to that theme."

Relevance Judgement: On this query, our goal is to retrieve movies with a spatial theme. To obtain good results, we need to use different fields: we can search for the *Science Fiction & Fantasy* genre, as well as space-related keywords on the textual reviews, movie title and description. Since we are only interested in movies available on Netflix, we need to apply a filter (*fq*) to the *available_netflix* field.

Query (q): "space sci-fi"

Filter query (fq): 'available_netflix: "True"'

Query fields (qf): *genres*, *original_title*, *movie_info* and *review_content*

Table 3: Weights applied to IN1

Field	Weight
genres	30
original_title	10
movie_info	50
review_content	20

Results: The first two systems resulted in a poor information retrieval performance, as can be seen in table 4. However, it is noticeable that, when applying boosts (table 3) to certain fields, the precision (P@10) increased up to 92%. This means that relevant results appear much sooner on system 3 (schema with boosts). In this query, we can prove that weights have a high influence over the relevance of the first 10 documents retrieved. The precision-recall curve for this information need is displayed in figure 12 (*Attachments*). From this curve, we can infer that the system with the weights filter is the one with the best performance. When the weights are not applied the precision falls abruptly after the first few documents.

Table 4: IN1 Results

	Schemaless	Schema	Boosted
P@10	0.361385	0.266667	0.92
AvP	0.4	0.2	0.6

B. Movies about slavery (IN2)

Information need: Movies that portray slavery and the different points of view over time

User story: "As a History enthusiast, I want to know more about the slavery topic"

Relevance Judgement: On this query, our goal is to retrieve movies that touch on the slavery subject. In order to obtain good results, we made our search by using slavery-related keywords on the movie title, movie info and review content.

Query (q): slavery slave

Query fields (qf): *original_title*, *movie_info* and *review_content*

Table 5: Weights applied to IN2

Field	Weight
original_title	10
movie_info	50
review_content	30

Table 6: IN2 Results

	Schemaless	Schema	Boosted
P@10	0.4	0.4	0.5
AvP	0.327563	0.325262	0.416569

Results: By analyzing the previous table, we noticed that the precision had a slight increase (about 10%) with boosting, relatively to Schema. However, Schema was insignificant compared with *Schemaless* as the values were identical in both fields. Bearing this in mind, as we wanted to avoid the keywords passed in *the query (q)* to point to non-relevant or out-of-context movies, the movie_info field had the highest boost value to ensure that our results were suited to the input. The precision-recall curve for all the different systems relative to this information need can be consulted in figure 13 (*Attachments*). This curve shows us that the systems have similar performance - the system with weights has better precision for the first few documents. When the recall is higher, the precision falls (as expected).

C. Emotive movies about World War II (IN3)

Information need: Emotive movies about World War II with high IMDB rating.

User Story: "As a history enthusiast, I want to find emotive movies about World War II with a high IMDB rating."

Relevance Judgement: On this query, we are expecting to retrieve documents related to emotive movies set in World War 2 with a high IMDB rating. To obtain movies with a high average score on IMDB, we apply a filter (*fq*). To retrieve movies considered emotive, the textual reviews are very important, since they contain the users' subjective thoughts on the respective movie (e.g a user that found the movie touching and emotive will likely express his feelings in this field).

Query (q): "World war II" emotional

Filter query (fq): "mean_vote_imdb: [8.0 TO 10.0]"

Query fields (qf): *movie_info* and *review_content*

Table 7: Weights applied to IN3

Field	Weight
movie_info	20
review_content	40

Results: For this Information Need, we can see in table 8 that the *Precision @ 10* was the same for all the 3 systems, as opposed to what happens in the remaining information needs. The chosen keyword ("emotional") is a keyword mentioned in several non-relevant documents. These misleading reviews make the system 3 results similar to the other systems - the Precision at 10 is equal for the 3 systems. When comparing the *Average Precision* metric, we can see that the *Boosted* system is the best one. followed by the system with *schema-only*. The precision-recall curve relative to this information need is identical for all 3 systems, as can be seen in figure 14 (*Attachments*). For higher recall values, the *schemaless* system ends up being the best.

Table 8: IN3 Results

	Schemaless	Schema	Boosted
P@10	0.5	0.5	0.5
AvP	0.564444	0.627778	0.68619

D. Movies about true crime stories (IN4)

Information need: Movies with a plot based on true crimes.

User story: "As a user interested in criminal stories based on real events, I want to find movies that describe such occurrences."

Relevance Judgement: On this query, our goal is to retrieve movies with a plot that is based on real crimes. In order to obtain good results, we searched for true crime-related keywords in movie information, as well as in review contents.

Query (q): "true crime story"

Query fields (qf): *movie_info* and *review_content*

Table 9: Weights applied to IN4

Field	Weight
movie_info	30
review_content	50

Results: The *schemaless* and *schema-only* systems both resulted in poor but similar results, as we can see in table 10. Nonetheless, it is clear that boosting certain fields (table 9) and applying phrase boost slop (3 words) - ***ps**: 3* - influences the results, since both the precision for the first 10 documents (P@10) and average precision (AvP) increased to approximately 50%. With this in mind, we can conclude that, in this query, boosting and *ps* have a visible influence over the relevance of documents retrieved. The precision-recall curve for this information need can be found in figure 15 (*Attachments*). Similar to what happens in IN1, the weights filter has a noticeable influence on the performance of the information retrieval. The *schemaless* system has an identical performance compared to the system with *schema-only*.

Table 10: IN4 Results

	Schemaless	Schema	Boosted
P@10	0.3	0.3	0.5
AvP	0.33899	0.349472	0.613563

D.Family movies to watch on Christmas (IN5)

Information need: Family-friendly movies to watch on the Christmas holiday.

User story: "As a user, I want to find an appropriate movie (suited for family) to watch with my family on Christmas eve."

Relevance Judgement: On this query, our goal is to retrieve movies that other users consider to be about Christmas or appropriate to watch during that time. Therefore, it is important to use textual reviews, as well as movies' titles and descriptions.

Query (q): "Christmas AND time"

Filter query (fq): 'genres:"Kids & Family"'

Query fields (qf): *original_title*, *movie_info* and *review_content*

Table 11: Weights applied to IN5

Field	Weight
movie_info	30
review_content	40

Results: In this need, every system showed similar results regarding *P@10*, as can see in table 12. On the other hand, when measuring the *AvP* metric we can see that the boosted system was the best one, obtaining great results (close to 90%). Bearing this in mind, we can conclude that the keywords were somewhat ambiguous on their own, which was taken care of by adding a phrase boost slop (5) to remove out-of-context keyword references.

Table 12: IN5 Results

	Schemaless	Schema	Boosted
P@10	0.7	0.7	0.7
AvP	0.696715	0.696715	0.893519

12 EVALUATION CONCLUSIONS

After collecting the results from the information needs experiments described, we can now compute the *Mean Average Precision* (MAP) of the 3 systems. Table 11 contains MAP values for each system.

Table 13: *MAP* Values for each system

Schemaless	Schema	Boosted
0.4655424	0.4398454	0.6419682

After evaluating the results from the Information Needs, we can see that system 3, which applies weights with a schema is the one with the best results. As expected, the weights improve the relevance of the documents retrieved. However, despite being the best system (with the highest *MAP*), it is still vulnerable to ambiguous keywords - keywords that exist in several non-relevant documents in the collection.

On the other hand, system 2 (*schema-only*) is not an improvement over the *schemaless* system. This means that the *schema* is something that could use improvement. It is also important to notice that the schema does not improve results for all information needs, since every information need is specific and may benefit from different *schemas*.

13 CONCLUSIONS AND FUTURE WORK

Our focus, in the first phase, was to choose a dataset with recent data and manipulate it to better understand its information and make it ready for the information retrieval phase. This goal was accomplished, despite having some problems with the merging and refinement process as well as some missing rows' values. The result is a reasonably large dataset (over 9,000 movies) with coherent data about movies, ratings and reviews.

After having the preprocessing stage done, we selected the *Solr* tool to develop an *Information Retrieval System*. We defined the indexing process, choosing what fields should be indexed and what custom field types should be added to the *Solr* schema. Having populated the collection, we chose 4 different information needs to evaluate the systems. This evaluation was done using the metrics *Precision @ 10* and *Average Precision*. The results obtained from these metrics allowed us to compare the systems used. We can conclude that applying the right weights to the correct fields generally improves the relevance of the documents retrieved. However, having general keywords increase the probability of retrieving non-relevant documents.

In future work, the goal will be to further improve our *Information Retrieval System*. This can be achieved by adding more than a single core or adding features and techniques to enhance the quality of the search system. Moreover, to improve the overall user experience, we plan on developing a graphical interface for the search system.

REFERENCES

[1] *Elasticsearch is the distributed, RESTful search and analytics engine at the heart of the Elastic Stack.* Last access: 15/12/2021. URL: https://www.elastic.co/pt/elasticsearch/.

[2] John Hunter. *A comprehensive library for creating static, animated, and interactive visualizations in Python.* Last access: 15/12/2021. URL: https://matplotlib.org/.

[3] Stefano Leone. *IMDB movies extensive dataset.* Last access: 15/12/2021. URL: https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset.

[4] Stefano Leone. *Rotten tomatoes movies and critic Reviews Dataset.* Last access: 15/12/2021. URL: https://www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset?select=rotten_tomatoes_critic_reviews.csv.

[5] *Listings of movies and tv shows on Netflix.* Last access: 15/12/2021. URL: https://www.kaggle.com/shivamb/netflix-shows.

[6] *Movies and TV Shows listings on Amazon Prime Video.* Last access: 15/12/2021. URL: https://www.kaggle.com/shivamb/amazon-prime-movies-and-tv-shows.

[7] *Movies and TV Shows listings on Disney+.* Last access: 15/12/2021. URL: https://www.kaggle.com/shivamb/disney-movies-and-tv-shows.

[8] Andreas Mueller. *Word Cloud is a data visualization technique used for representing text data.* Last access: 15/12/2021. URL: https://amueller.github.io/word_cloud/.

[9] *Pandas.* Last access: 15/12/2021. URL: https://pandas.pydata.org/.

[10] Kenneth Reitz. *An elegant and simple HTTP library for Python.* Last access: 15/12/2021. URL: https://docs.python-requests.org/en/latest/.

[11] *Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene.* Last access: 15/12/2021. URL: https://solr.apache.org/.

[12] *Spacy industrial-strength natural language processing in python.* Last access: 15/12/2021. URL: https://spacy.io/.

[13] *The Apache Software Foundation provides support for the Apache community of open-source software projects.* Last access: 15/12/2021. URL: https://lucene.apache.org/.

[14] Michael Waskom. *Seaborn is a Python data visualization library based on matplotlib.* Last access: 15/12/2021. URL: https://seaborn.pydata.org/.

[15] *Your machine learning and Data Science Community.* Last access: 15/12/2021. URL: https://www.kaggle.com/.

14 ATTACHMENTS

The next page contains the attachments for all mentioned figures throughout this paper.

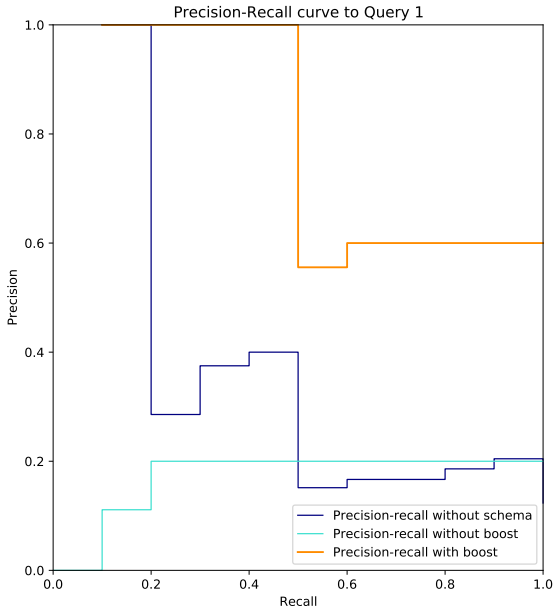

Figure 12: "Precision-recall curve for IN1"

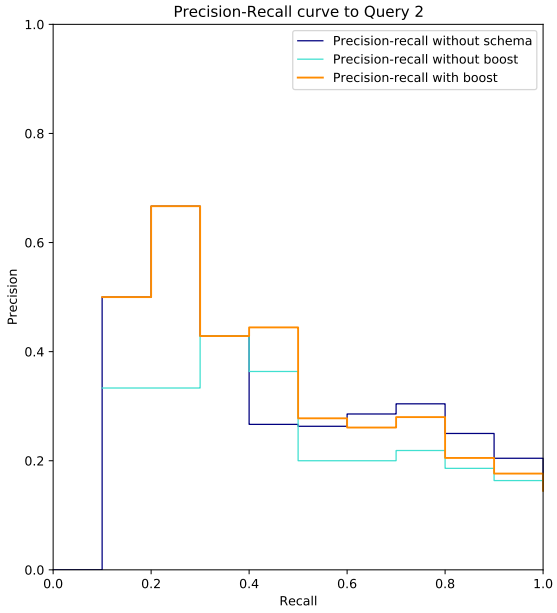

Figure 13: "Precision-recall curve for IN2"

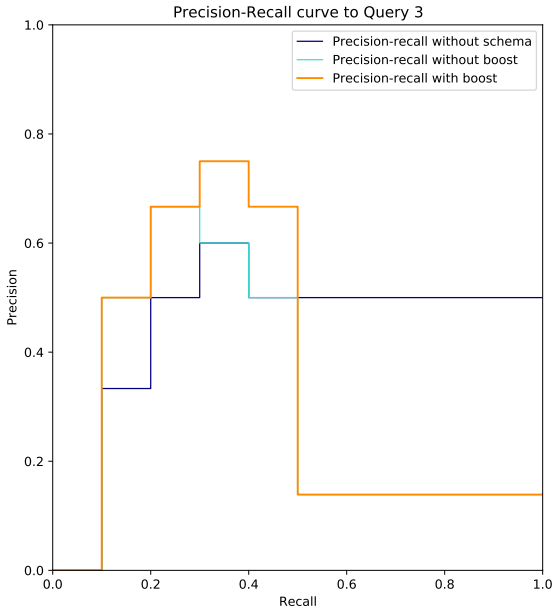

Figure 14: "Precision-recall curve for IN3"

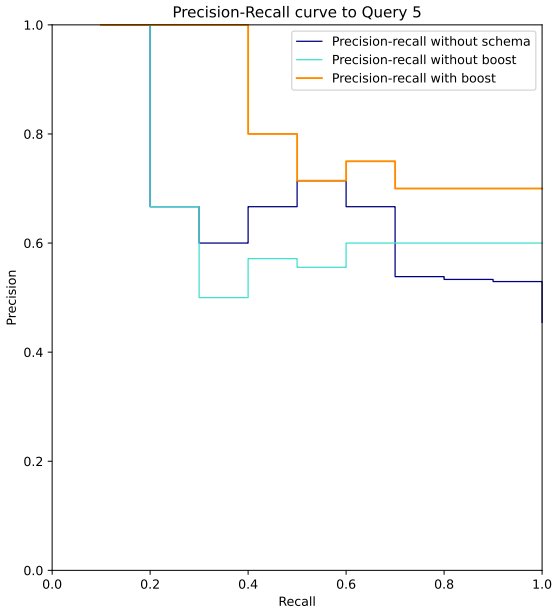

Figure 16: "Precision-recall curve for IN5"

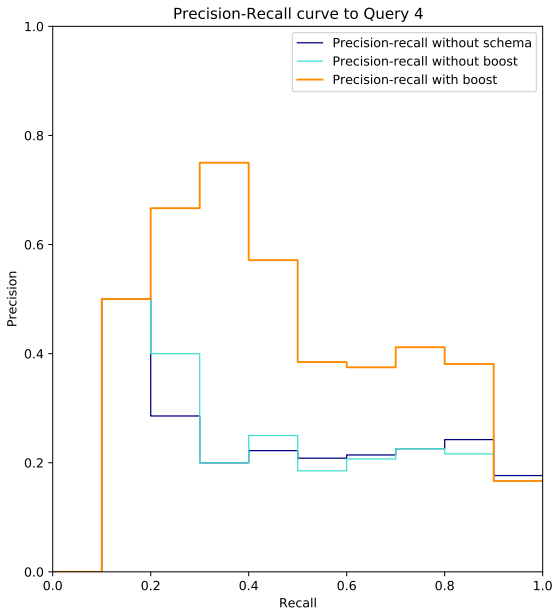

Figure 15: "Precision-recall curve for IN4"