IMDb Movies Processing and Retrieval Tool

Beatriz Aguiar, João Marinho, Margarida Vieira[*]

Abstract

This report explores the creation of an IMDb movies search engine. The goal is to create a system capable of answering non-trivial and ambiguous queries. Data preparation is the initial and fundamental phase. Its focus is on collecting data from IMDb database, processing it, and conducting an exploratory analysis for further understanding of the convenient search tasks. This is followed by the retrieval phase, in which the Solr tool is used. Indexing data and creating queries are the main focus. Lastly, some improvement attempts within the context of a more general movie search engine are made.

Keywords: datasets, data preparation, data analysis, information retrieval, search system

ACM Reference Format:
Beatriz Aguiar, João Marinho, Margarida Vieira. 2018. IMDb Movies Processing and Retrieval Tool. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Cinema has been a part of the entertainment industry for a long time and it creates a massive impact on people all over the world by offering people a momentary escape from reality, regardless of the type of movie being watched. For instance, the global pandemic in 2020 had a tremendous impact on this industry since individuals were quarantined and had no other source of cinematic content besides the internet and streaming platforms. But, misfortunes aside, movies allow us to identify with the characters on an emotional level - we see them as real people struggling with real problems and facing real challenges, we live vicariously through them and find in them our own personal fears and desires, hopes, and aspirations. Besides, movies offer a different perspective on the lives of people in other societies, providing insight into the lives and cultures of other people, and broadening our understanding of the world. For these reasons and a lot more, it made perfect sense to base our search system on this type of information.

This paper is split into three main parts. The first, **Data Preparation** - Section 2, focuses on the process of collecting, understanding, and refining the data in order to obtain a final version of the dataset. The second part, **Information Retrieval** - Section 3, begins with a description of the documents collection and their indexing process. Following the exploration of some previously identified search tasks through the formulation of queries with different levels of complexity, and the assessment of the retrieval tool results. Finally, in the third part, **Search System** - Section 4, is explored the feasibility of some hypotheses, aiming to improve the search system.

2 Data Preparation

The goal of the first task is to prepare and explore datasets, which may suffer from extraction actions such as crawling or scraping.

2.1 Data Selection

The provided guidelines shepherd the pursuit of rich and ambiguous datasets that granted both information quantity (in the thousands range) and quality. Thus, movies came across as the ideal theme of choice.

Google Dataset Search was our first approach to the search for datasets, however, after a generic analysis, we considered the available data to be quite simple, containing only one table, while lacking textual features.

Given IMDb's high demand and popularity, it seemed to be the ideal dataset source. Its API did not only provide 7 TSV files, but it also granted reliable and updated content, with each table containing different media-related data up to millions of entries. Such information covered most of the common movie attributes (e.g. duration, genre), titles, directors, actors, and ratings.

Subsets of IMDb data are available for access to customers for personal and non-commercial use. You can hold local copies of this data while being subject to their terms and conditions. Information courtesy of IMDb [1]. Used with permission.

2.2 Data Processing

As demonstrated in figure 1, the original datasets underwent through a thorough processing, following an ETL (extract-transform-load) like pattern. To collect the data, the makefile

[*] Beatriz Aguiar, up201906230@up.pt; João Marinho: up201905952@up.pt; Margarida Vieira, up201907907@up.pt.

Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1] (https://www.imdb.com)

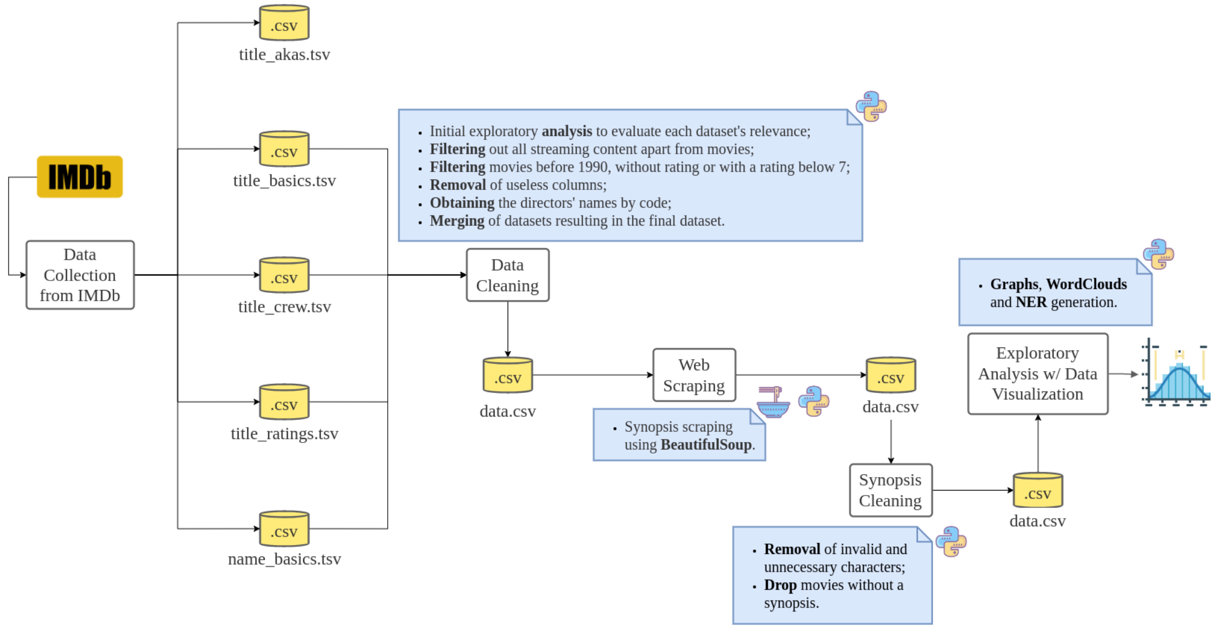

.csv
title_akas.tsv
.csv
title_basics.tsv
IMDb
Data Collection from IMDb
.csv
title_crew.tsv
.csv
title_ratings.tsv
.csv
name_basics.tsv
Initial exploratory **analysis** to evaluate each dataset's relevance;
Filtering out all streaming content apart from movies;
Filtering movies before 1990, without rating or with a rating below 7;
Removal of useless columns;
Obtaining the directors' names by code;
Merging of datasets resulting in the final dataset.
Data Cleaning
.csv
data.csv
Web Scraping
Synopsis scraping using **BeautifulSoup**.
.csv
data.csv
Synopsis Cleaning
Removal of invalid and unnecessary characters;
Drop movies without a synopsis.
.csv
data.csv
Graphs, **WordClouds** and **NER** generation.
Exploratory Analysis w/ Data Visualization

Figure 1. Pipeline

starts by running a series of commands that download the zip files from the IMDb API and unzip the latter to the data folder.

An initial exploratory analysis was conducted on each dataset to understand its relevance and features importance. The *title_akas.tsv* dataset, for e.g., turned out to be uninteresting as the only feature we were interested in, the language of the movie, had 6 not-null values in a 33M entries dataset. On top of that, we concluded that the datasets regarding the principal crew members and episodes of the streaming content were also irrelevant to our search engine.

In terms of data preparation, we started by filtering the dataset *title_basics.tsv*, leaving only entries corresponding to movies released after 1990. We opted to do this not only for efficiency purposes, by reducing the total number of entries, but also due to the minimal value it added to the dataset - fewer individuals will likely search for older data.

However, the dataset still comprised hundreds of thousands of entries, which was far more data than necessary and would increase web-scraping latency. For this reason, we decided to discard movies with average ratings below 7, always aiming to keep the most relevant data possible. The final dataset now contained around seventy thousand entries, a considerably good dataset dimension.

Posteriorly, we moved on to merge the other datasets with the main movie dataset.

At this point, our dataset's features consisted in an id - *tconst*, the original movie name - *originalTitle*, a boolean indicating whether it is an adult movie or not - *isAdult*, the release year - *startYear*, the duration of the movie - *runtimeMinutes*, its genres, the average rating, the number of votes, and, finally, the directors, a string of ids referring to the crew's dataset.

The next step was to match the directors' string of ids to the corresponding real names and convert it to a string of names.

For the most important movie's piece of information, the synopsis, we performed web-scraping on both Wikipedia and IMDb's pages, using Python's *BeautifulSoup* package. This textual feature did not, however, suffer from any processing other than cleaning up irrelevant characters.

Finally, we chose to standardize the features' names to enhance the posterior visual analysis carried out - *startYear* changed to *year*, *runtimeMinutes* to *duration*, and *averageRating* to *avgRating*.

2.3 Conceptual Schema

Figure 2 showcases how we can structure each piece of information in a Class based model.

Movies can be categorized into Adult and Non-Adult ones, having one or more genres and multiple directors.

2.4 Characterization

To better understand the data in our hands, we developed a Python Notebook to plot general information about the dataset. The more information we have, the better the decisions will be made throughout the development of the project.

By taking a look at the obtained graphics, some conclusions can be easily drawn.

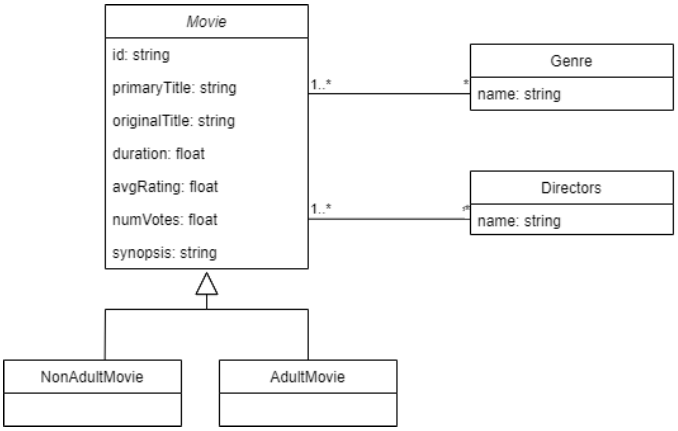


Figure 2. Conceptual Model

First, the number of movies released per year (figure 3) has increased in the last decade, meaning that more information can be drawn from more recent years. This increase is probably due to the latest technological advances.

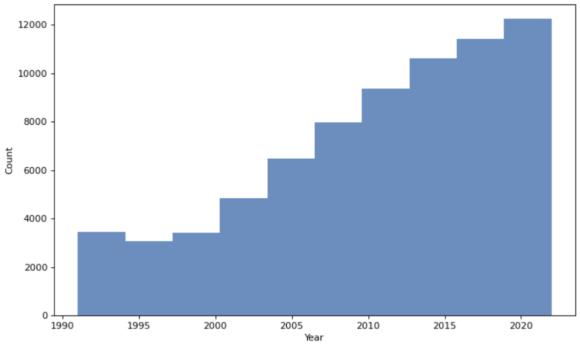


Figure 3. Number of movies per year

The second plot (figure 4) shows that only a small minority of films were able to get ratings above 8, with most films scoring between 7 and 8. However, it is important to keep in mind that the final dataset did not include any films with an average rating of less than 7.

When it comes to genres (figure 5), Documentary and Drama seem to be the most popular ones by an obvious margin, with Comedy, Biography, and Romance completing the top 5.

Since the average rating for practically every genre falls within the same range, no meaningful conclusions can be drawn about the average rating per genre (figure 6).

The following plot, *Rating per number of votes* (figure 7), demonstrates that just a minor number of individuals give a rating above 8, leading us to believe that ratings above this threshold are simply the consequence of fewer votes when compared to ratings between 7 and 8, which usually receive the majority of votes.

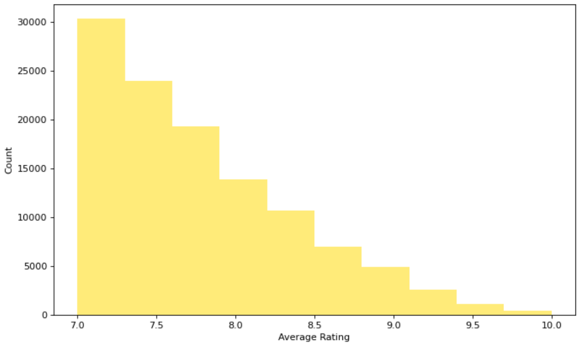


Figure 4. Number of movies per rating

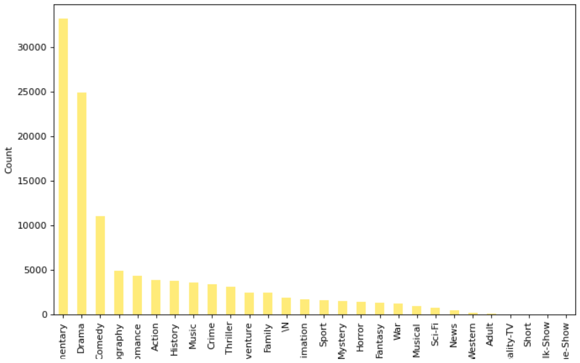


Figure 5. Number of movies per genre

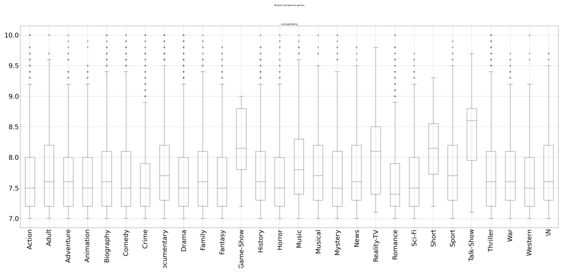


Figure 6. Average rating per genre

We also decided to plot the number of adult vs. non-adult movies (figure 8).

Finally, we plotted a WordCloud for 3 of the top 5 genres - Documentary, figure 9, Biography, figure 10, and Romance, figure 11.

A further step towards information extraction was conducted, Named Entity Recognition (NER), in the newest feature attained, the synopsis. This process mainly tries to locate and classify entities such as characters' names and/or specific

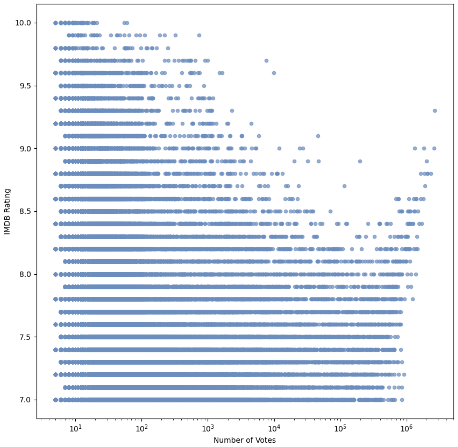


Figure 7. Rating per number of votes

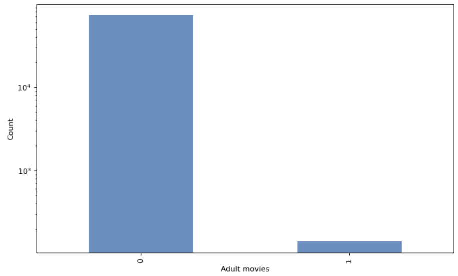


Figure 8. Number of movies per adultness

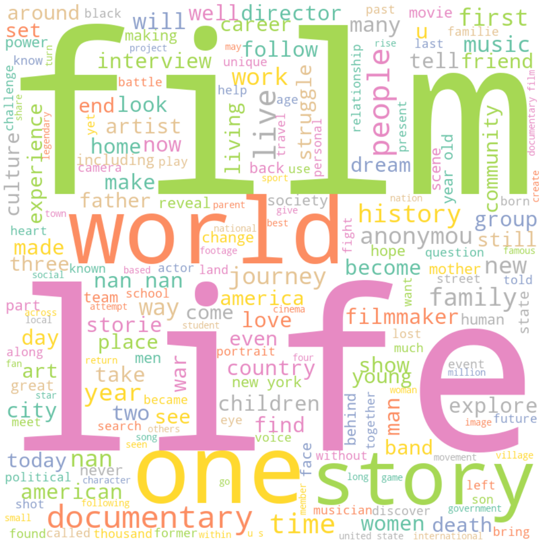


Figure 9. Documentary movies' synopsis wordcloud

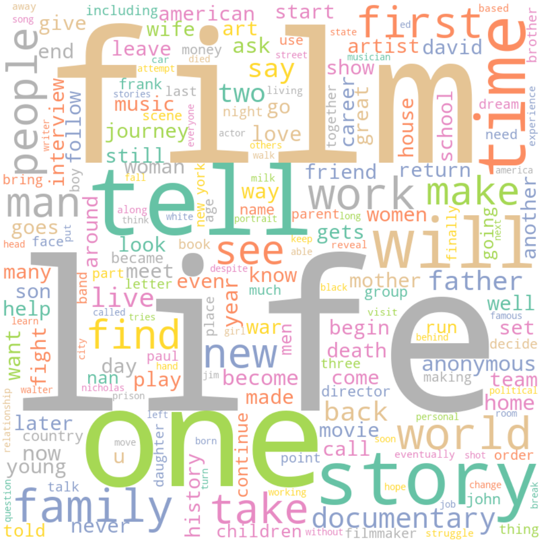


Figure 10. Biography movies' synopsis wordcloud

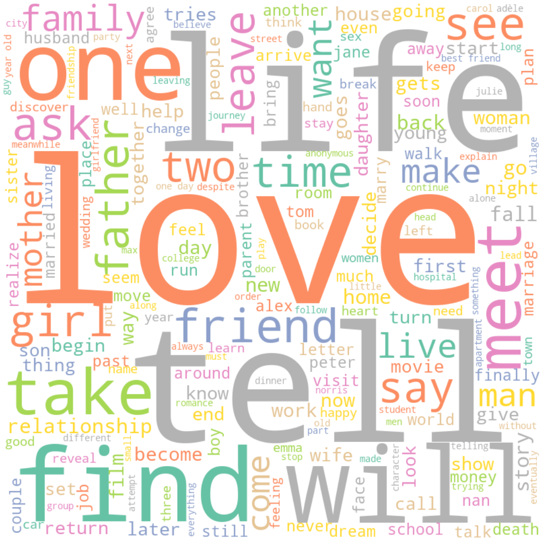
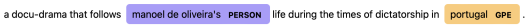


Figure 11. Romance movies' synopsis wordcloud

places. Figure 12 is an example of this, applied to a biography synopsis.

In addition to NER, it was also performed a Keyword extraction analysis. For the previous NER example, the resulting keyword extraction list was:



a docu-drama that follows manoel de oliveira's PERSON life during the times of dictatorship in portugal GPE .

Figure 12. Example of a NER biography synopsis.

1. 'drama follows manoel oliveira'
2. 'follows manoel oliveira life'
3. 'oliveira life times dictatorship'

4. 'life times dictatorship portugal'
5. 'manoel oliveira life'
6. 'dictatorship portugal'

2.5 Prospective Search Tasks

- Family Movies
- Movies about the space
- Romance teen movies
- Movies to watch in your twenties
- Life-changing movies

3 Information Retrieval

Information Retrieval refers to the activities of obtaining information resources (usually in the form of textual documents) from a much larger collection, which are relevant to a user's information need, usually expressed as a query. It is the science of searching for information in a document, searching for documents themselves, and also searching for the metadata that describes data, and for databases of texts, images or sounds.

The goal of the second task is to implement and use an Information Retrieval Tool - Solr - on the final dataset, assembled in Section 2, and explore it with free-text queries.

3.1 Collection and Indexing

In the system, a movie defines a document. As there was no other type of documents, the queries were made upon a single collection. The recommended *Solr* [2] tool was utilized to achieve this.

Formerly, we described how the data sources were merged into a single database-like structure. However, some refinements could be made to improve our experience with the said tool. These included:

- Splitting the genres and directors fields into a list of strings.
- Converting the end-resulting CSV file to JSON format, allowing for easier use of multivalued fields.

Regarding the indexing process, two schemas were created, a basic one and an improved version. Both contain the same fields, and field types, differentiating the tokens and filters applied.

The first step was to determine which features would be relevant for the user to query. These features compromised the movie's primary and original titles, release year, genres, average rating, directors, and synopsis. Although not being indexed, all the other features are stored since they provide some other interesting aspects about movies (Table 1).

Secondly, we had to specify which tokenizer and filters would be applied at index and query time to the more rich textual fields (*textualField*). For the basic schema definition (Table 2), this more complex pre-processing employed the ***StandardTokenizerFactory*** class, and two filters:

- ***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.
- ***LowerCaseFilterFactory*** which converts any uppercase letters in a token to the equivalent lowercase token.

Field	Type	Indexed
tconst	string	No
primaryTitle	textualField	Yes
originalTitle	textualField	Yes
isAdult	bool	No
startYear	int	Yes
runtimeMinutes	float	No
genres	textualField	Yes
averageRating	float	Yes
numVotes	int	No
directors	nameField	Yes
synopsis	textualField	Yes

Table 1. Schema fields.

Name	Tokenizer	Filters
textualField	StandardTokenizerFactory	ASCIIFoldingFilterFactory LowerCaseFilterFactory

Table 2. Basic schema custom field types.

While the enhanced schema relied on the ***LowerCaseTokenizerFactory***, which behaves much like the combination of the Standard Tokenizer and the Lowercase Filter, with the following filters:

- ***StopFilterFactory*** which stops (by discarding) the analysis of, tokens that are on the default stop words list.
- ***EnglishMinimalStemFilterFactory*** which stems plural English words to their singular form.
- ***EnglishPossessiveFilterFactory*** which removes singular possessives (trailing 's) from words.
- ***PorterStemFilterFactory*** which applies the Porter Stemming Algorithm for English.

The above-mentioned field types apply the same provisions at both the index and query time.

We have also used a custom type for the list of director names since it does not make sense to apply most of the filtering as in rich text features (Table 3).

[2] (https://solr.apache.org/guide/8_1/)

Name	Tokenizer	Filters
textualField	LowerCaseTokenizerFactory	ASCIIFoldingFilterFactory StopFilterFactory EnglishMinimalStemFilterFactory EnglishPossessiveFilterFactory PorterStemFilterFactory
nameField	LowerCaseTokenizerFactory	ASCIIFoldingFilterFactory

Table 3. Enhanced schema custom field types.

3.2 Retrieval

After the indexing process is complete, the next step was to decide how to retrieve the information from the indexed documents.

Three different information needs were defined, based on the prospective search tasks previously defined in Subsection 2.5. For each information need, the motivation for the topic, a brief description, and the basic and boosted queries and correspondent parameters are provided.

3.2.1 Information Need 1 - Movies about the space.

With this information need, we intended to retrieve all movies about the space.

Considering the ambiguity of the word "space", the query was not as straightforward as it seemed when we first thought of it. There were other words that directly related more to the topic, so while we had to include the word "space" in the query, we couldn't give it too much relevance.

Basic Query

- **query**: space, astronaut, galaxy, planets
- **query fields**: originalTitle, primaryTitle, synopsis

Boosted Query

- **query**: space, astronaut, galaxy, planets
- **query fields**: originalTitle^1.5, primaryTitle^1.5, synopsis^2.0
- **boost query**: genres:sci-fi^3.0, synopsis:stars

3.2.2 Information Need 2 - Romance teen movies.

After analysing the synopsis of various romance teen movies which are suggested as the most popular movies of the said type [1], we concluded that these are often described with the words love, crushes, teen, college, and some others. This led us to come up with a set of keywords to compose the query.

As for genres, despite many of these movies being characterized as *romcoms* (romantic comedies), we chose to give more importance to romances and dramas, in the boosted query.

Basic Query

- **query**: romance, teen, crush, heart-break, in love, high-school, college, friends, friendship, campus, gossip, passion, attraction
- **query fields**: originalTitle, primaryTitle, synopsis

Boosted Query

- **query**: romance, teen^3.0, crush, heart-break^3.0, in love^2.0, high-school^2.0, college, friends, friendship, campus, gossip, passion, attraction
- **query fields**: originalTitle^1.5, primaryTitle^1.5, synopsis^4.0
- **boost query**: genres:drama, genres:romance, synopsis: crush^2.0

3.2.3 Information Need 3 - Movies to watch during Christmas Eve.

The best memories from our childhood Christmases (aside from the toys under the tree) are probably wrapped up in the family coming together to watch some classic Christmas movies. With this information need, we intended to find just the perfect movies to watch during that season.

The strategy used to put together this query, especially its keywords, was fairly similar to the one used for the previous query. However, in this case, and for what we think to be obvious reasons, we chose to boost documents where the term "christmas" was featured in the movie's title or synopsis.

Basic Query

- **query**: christmas^2.0, santa^2.0, snow, elf, rodolf, festive, claus, merry, holiday, candy cane, christmas tree, eve, advent
- **query fields**: originalTitle, primaryTitle, synopsis

Boosted Query

- **query**: christmas^2.0, santa^2.0, snow, elf, rodolf, festive, claus, merry, holiday, candy cane, christmas tree, eve, advent
- **query fields**: originalTitle^1.5, primaryTitle^1.5, synopsis^2.0
- **boost query**: primaryTitle: christmas^5.0, synopsis:christmas^3.0

3.3 Evaluation

To obtain a more objective judgement of our system's performance and compare different versions of the same base system, we strive to evaluate the obtained results systematically. We will then compare normal and boosted query results with a basic and an enhanced schema version.

Precision and Recall are two well-known measurements used to quantify retrieval effectiveness. Precision is the fraction of retrieved documents relevant to the query, i.e., the ratio between the number of correct results and the total number of retrieved results. Recall, on the other hand, is the fraction of relevant instances which were retrieved.

The above-mentioned metrics tend to vary inversely: higher precision is often coupled with a lower recall. Therefore, they are often presented together in the form of precision-recall curves. These plots allow us to observe the trade-off between the metrics at certain thresholds and measure the system's quality when acknowledging the area under the curve, hence why we chose to use them.

Other metrics used are the Precision at N (P@N) and the Average Precision (AP). For the Precision, N assumes values 10 and 15, while for the AP we calculate both the Average Precision@10 and globally. Finally, the Mean Average Precision (MAP), which averages the global APs of each query, provides a better comparison for different systems.

A typical evaluation approach relies on test collections, i.e., a series of topics (information needs), a set of documents to be searched called the corpora, and relevance judgments indicating which documents are relevant for which topics.

Given the extensive dataset and the need to know which movies were relevant for each query, resulting in a significantly time-consuming process, we opted to extract test collections (sample of 300 documents) from the initial dataset for evaluation purposes.

Firstly, by analyzing the results retrieved from Solr, we could, with a certain margin of error, understand the ratio of relevant movies for each query.

Secondly, we began by creating a smaller dataset with the help of the existing IMDb Information Retrieval Tool [3]. By intersecting its results with our documents, we extracted a set of relevant records whose number would still represent the percentage of relevant documents in the original dataset.

Finally, all information needs were obtained and evaluated using this sample subset to minimize the generalization error.

3.3.1 Information Need 1 - Movies about the space.

The evaluation results can be seen in figures 13 and 14 and tables 4 and 5.

The results were very satisfactory, having an Average Precision always above 0.95 and a Recall and Precision at 10 of 1.0. Our top results were relevant and all the relevant documents were retrieved.

The boosted query did improve the results when compared to the basic one. Despite the latter having a very high Average Precision, the applied boosts proved to be effective but not in a very noticeable way.

When comparing the results of the basic schema (figure **??** and table 4) to the enhanced schema (figure **??** and table 5), we found it surprising that the enhanced schema showed a worse Average Precision.

After manually analysing the results, we concluded that this deterioration was due to the application of the *English-MinimalStemFilterFactory* in the enhanced schema. For example, an irrelevant movie was retrieved because the 'spaces' word, contained in the synopsis ("...Santa can't move quickly or through tight spaces like he does, so the kids are forced to help him..."), was converted to 'space' and, therefore, increased the document's score.

Nevertheless, we observed that the order of the results was subjectively better when using the enhanced schema and the irrelevant documents retrieved had the lowest score. This may allow for future filtering according to the score given by the used tool, with the imposition of a lower bound for the said attribute.

Metric	Basic	Boosted
Global Average Precision	0.989733	0.998462
Average Precision at 10	1.0	1.0
Precision at 10 (P@10)	1.0	1.0
Precision at 15 (P@15)	1.0	1.0
Individual Assessment	RRRRRRRRRR	RRRRRRRRRR

Table 4. Space Movies Basic Schema Metric Results.

Metric	Basic	Boosted
Global Average Precision	0.971166	0.984371
Average Precision at 10	1.0	1.0
Precision at 10 (P@10)	1.0	1.0
Precision at 15 (P@15)	1.0	1.0
Individual Assessment	RRRRRRRRRR	RRRRRRRRRR

Table 5. Space Movies Enhanced Schema Metric Results.

3.3.2 Information Need 2 - Romance teen movies.

The evaluation results can be seen in figures 15 and 16 and tables 6 and 7.

In this information need, the improvements of the schema are more visible and the boosting raises the Precision, reaching an Average Precision of 1.0 with the enhanced schema. These results are extremely good, since, for our best environment, Solr retrieves only relevant documents, which is ideal in a search engine.

Metric	Basic	Boosted
Global Average Precision	0.535311	0.915972
Average Precision at 10	0.891723	0.915972
Precision at 10 (P@10)	0.7	0.8
Precision at 15 (P@15)	0.6	0.53
Individual Assessment	RRRRNNRRRN	RRRRRNNRRR

Table 6. Romance Teen Movies Basic Schema Metric Results.

[3] https://www.imdb.com/search/keyword/

Metric	Basic	Boosted
Global Average Precision	0.558238	0.906796
Average Precision at 10	0.895685	0.906796
Precision at 10 (P@10)	0.8	0.8
Precision at 15 (P@15)	0.66	0.53
Individual Assessment	RRRNRRRRNR	RRRNRRRRRN

Table 7. Romance Teen Movies Enhanced Schema Metric Results.

3.3.3 Information Need 3 - Movies to watch during Christmas Eve.

The evaluation results can be seen in figures 17 and 18 and tables 8 and 9.

In this information need, neither the boosting nor the schema have a major impact on the Precision. However, there is an evident improvement when evaluating the metric *Precision at 10*.

This means our boosted query, besides retrieving more relevant documents, scores them higher than non-relevant ones. For example, 60% of irrelevant documents are in the top 10 results in the basic query. We found this improvement to be very useful due to the importance of top results in a search engine.

One of the other top search engines after Google, Bing, reports that websites on the top get 42% of the traffic; the second gets 11% and the third only gets 8%. So, it clearly indicates why your website needs to be on top. [4]

Metric	Basic	Boosted
Global Average Precision	0.688843	0.935947
Average Precision at 10	1.0	1.0
Precision at 10 (P@10)	0.4	0.8
Precision at 15 (P@15)	0.53	0.8
Individual Assessment	RRRRNNNNNN	RRRRRRRRNN

Table 8. Christmas Movies Basic Schema Metric Results.

Metric	Basic	Boosted
Average Precision	0.783282	0.967493
Average Precision at 10	0.870833	0.988889
Precision at 10 (P@10)	0.6	0.9
Precision at 15 (P@15)	0.73	0.86
Individual Assessment	RRRRNNNRNR	RRRRRRRRNR

Table 9. Christmas Movies Enhanced Schema Metric Results.

Overall, we can conclude that both the boosted queries and the enhanced schema perform as desired, by pushing the most relevant documents to the top and by retrieving fewer irrelevant documents.

4 Search System

The results from Section 3 were satisfying, especially those from the combination of the enhanced schema and the boosted queries. For example, considering the Average Precision for the top 10 results, we managed to achieve a value of over 0.85 for every query. However, these outcomes can be misleading if the search engine is used more naturally since we input particular Solr parameters (boosts) for each query.

The goal of the third task is to explore innovative approaches, aiming to improve the search system from a more general perspective. With this in mind, we will compare different system performances using the Mean Average Precision (MAP) as the main evaluation metric against two distinct hypotheses and some minor improvements.

4.1 Smaller Improvements

4.1.1 Filters.

Regarding the present search system, aside from over-fitting the queries, improving the schema is the most optimistic and safe option. After deep analysis and search, we removed some useless and redundant filters, such as the *EnglishPossessiveFilterFactory* and the *EnglishMinimalStemFilterFactory* since we already applied the *PorterStemFilterFactory*. We have also decided to explore the *BeiderMorseFilterFactory* filter for name fields, in our case the *directors* feature.

4.1.2 Synonyms.

Another attempt to improve and generalize the search system was to use synonyms in Solr. We created a *synonyms.txt* file containing some relevant synonyms for the previously mentioned information needs, aiming to analyze the impact on the results. For this, the *SynonymGraphFilterFactory* filter was applied to our *textualField* at the indexing level.

4.1.3 Results.

As expected, removing the redundant filters had no impact on the results. Despite the *BeiderMorseFilterFactory* constituting an improvement to our search system, it had no impact on our queries as the *directors* field is not used.

The use of the synonyms, however, impacted the results negatively, lowering the Global Average Precision and the Christmas and Space movies’ precision. Therefore, we opted to discard this filter from the schema.

4.2 Hypothesis 1 - Signal Exploration

In the context of a movie search engine, the results matching the user query are not the only relevant aspect. Other facets, such as popularity or release dates, should be taken into consideration.

In an attempt to improve the user experience while maintaining the relevancy of the documents retrieved. We decided to explore the impact of independent query signals, such as the *averageRating* and *numVotes*, on our end results.

4.2.1 Process.

The process started off by calculating an increment value to be added to the original document's score. For this, we created a new feature regarding the film's popularity which was computed by multiplying the movie's average rating by the number of votes.

Since we want the new field to have a partial weight in the final score, we applied *Min Max Scaling*, with a minimum value of 0 and a maximum of 2. These values were reached through the study of our initial score ranges so that an increment to the latter would not make substantial changes to our retrieved documents order.

Ultimately the increment was added to the original Solr score and the documents were re-ordered.

4.2.2 Results.

Overall the results were very similar to the ones from the initial system, as we can see from the small MAP difference between both (Table 10).

Metric	Basic	Signal Exploration
Mean Avg Precision	0.798942	0.796801

Table 10. MAP results - Signal Exploration.

Although one might think no major improvements were made with the hypothesis in question, the truth is that there is a higher chance that better-ranked and more popular movies appear before poorly-ranked and not well-known ones, which is an overall improvement for the user experience.

On the other hand, the hypothesis has a greater impact when the query results have very similar scores, in which case the increment makes the difference in the ranking process.

4.3 Hypothesis 2 - Learning To Rank

Learning to Rank (LTR) is a class of techniques that apply supervised machine learning to build ranking models for information retrieval systems. It differs from traditional machine learning because its primary focus is to solve ranking issues rather than prediction issues. LTR is used to re-rank the top N retrieved documents using trained machine learning models, hoping that such models can make more nuanced ranking decisions than standard ranking functions. [2]

4.3.1 Configuration.

The Solr's *LTR* contrib module allows you to configure and run machine-learned ranking models, although some requirements must be followed for its good functioning.

The first step is to update the existing *solrconfig.xml* file with the listed configurations from the *Solr LTR tutorial page* [3].

Lastly, Slor must be started with the *LTR* plugins enabled, since they come disabled by default.

4.3.2 Features.

The way a ranking model computes the scores to rerank documents is done by using one or more of the three types of inputs:

- parameters that represent the scoring algorithm
- features that represent the document being scored
- features that represent the query for which the document is being scored

In this subsection, we will be focusing on the last two topics.

Features can be described as a value, a number, that represents some quantity or quality of the document being scored or of the query for which documents are being scored [3].

For the success of the ranking process, we conducted an exploratory analysis of the current dataset. This, allowed us to decide which features should be kept or extracted, i.e. the ones that revealed significance to the identification of a document.

The final set of features included the **rawYear** representing the startYear of a movie,**queryMatchTitle** that matches the query with the primary title of a movie, the **queryMatchSynopsis** that matches the query with the movie synopsis, the Solr's original score and the increment mentioned in section 4.2.

4.3.3 Model.

Solr *LTR* contrib module supports some generalized forms of models, such as Linear, Multiple Additive Trees and Neural Network models. Our approach was based on the simpler of the three, Linear, with the suggested Solr specification, **RankSVM** [5].

RankSVM uses a pairwise approach for ranking documents, i.e., ranks documents based on relative score differences and not for being close to the label, as the pointwise approach does. The latter possesses the disadvantage of compromising ordering which is a major drawback when considering some metrics.

Regarding the weights given to each feature, we must train our model beforehand (offline), and end the process by uploading the model's configuration JSON file to Solr.

Feature	Weight
rawYear	0.034254838
increment	0.67846161
queryMatchTitle	0
queryMatchSynopsis	-0.057850242
originalScore	0.28489178

Table 11. Features used with LTR and respective weights

4.3.4 Results.

Even though the use of *LTR* seemed promising, it revealed disappointing results, by lowering the MAP to 0.02. The only metric that was increased with this hypothesis was the Global Average Precision for romance teen movies, however, the Precision at 10 and at 15 decreased significantly.

An overall overview of the results can be seen in figures 22, 23 and 24, which contain a global comparison of both hypotheses for the Christmas, romance and space movies, respectively.

A possible reason for the model's results being worse is the number of features provided to the machine learning model, which were fairly small. Even though we attempted to provide more features, specifically the length of some textual fields and all the other query-independent numerical fields, this resulted in a massive training execution (+10 hours) time which precluded us from obtaining results.

Metric	Basic	LTR
Mean Avg Precision	0.798942	0.774413

Table 12. MAP results - Learning to Rank.

10.

5 Conclusion

This paper describes, in detail, the process of creating an IMDb movies search engine. The first part consisted of the data selection and processing, the whole procedure is represented as a pipeline in figure 1. Its importance resides in creating a final dataset, cleaned and properly analyzed, to be used as the main data source of the final search engine. In the second part, we implemented and used the Information Retrieval Tool, Solr, with the generated dataset. Regarding the retrieval process, two schemas for indexing the dataset were created and two queries, for each information need, were constructed. The implemented system was evaluated by comparing the use of each schema in both queries. Lastly, to finalize the search system, we implemented and studied some techniques to improve the quality of the search results. Our focus was on altering the schema, using signals and the Learning to Rank algorithm.

References

[1] 2022. Romance Teen Movies Keyword Search. shorturl.at/hmvHM

[2] Nikhil Dandekar. 2016. Intuitive explanation of Learning to Rank (and RankNet, LambdaRank and LambdaMART) | by Nikhil Dandekar | Medium. https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418

[3] Apache Software Foundation. 2021. Learning To Rank | Apache Solr Reference Guide 8.10. https://solr.apache.org/guide/8_10/learning-to-rank.html

[4] Ramona Sukhraj. 2017. How Important Is a Top Listing In Google? (And Why?). https://www.impactplus.com/blog/important-top-listing-google

[5] Joachims Thorsten. 2009. SVMrank | Support Vector Machine for Ranking. https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

A Images

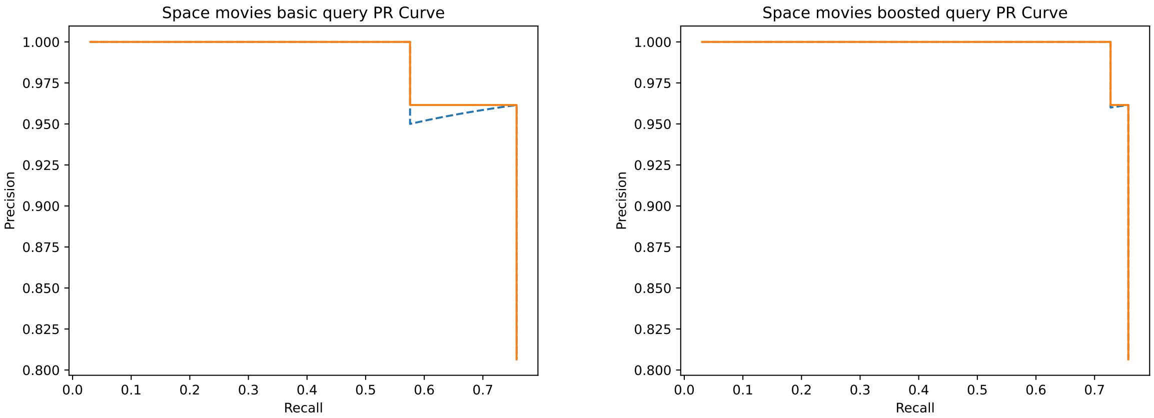


Figure 13. Space Movies PR Curves - Basic vs Boosted

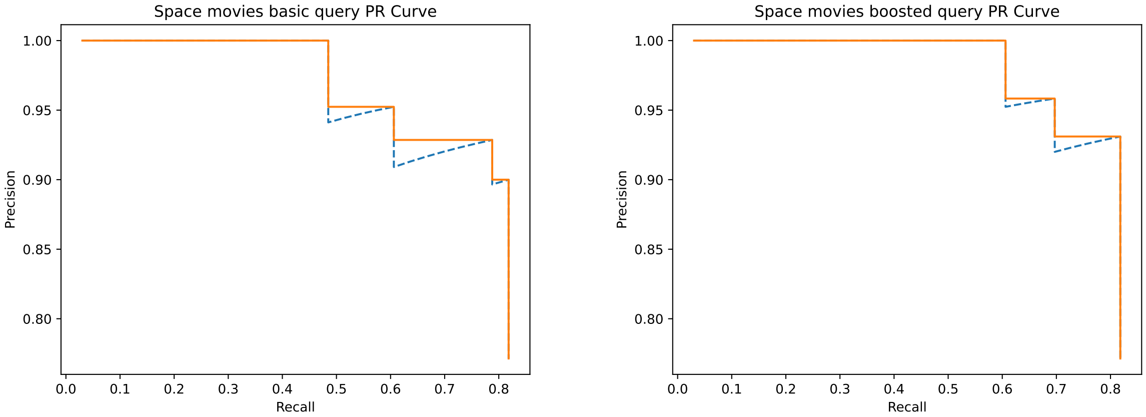


Figure 14. Space Movies Enhanced Schema PR Curves - Basic vs Boosted

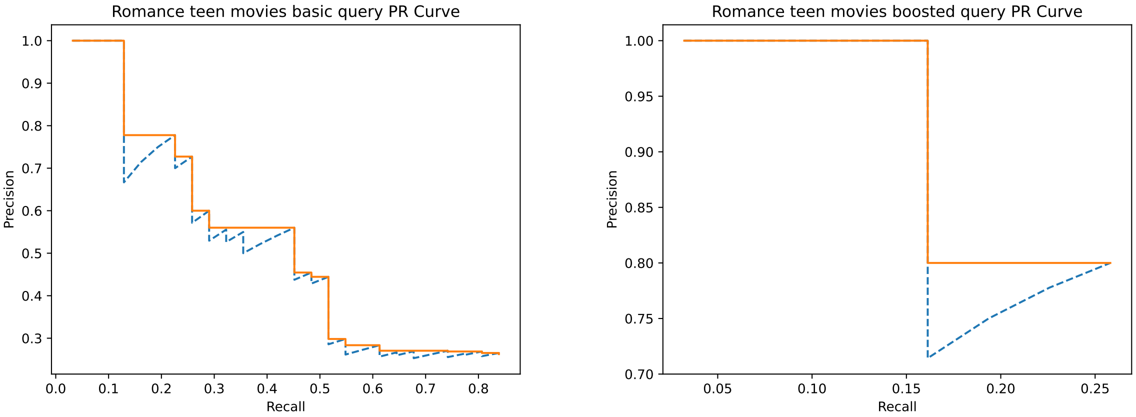


Figure 15. Romance Teen Movies PR Curves - Basic vs Boosted

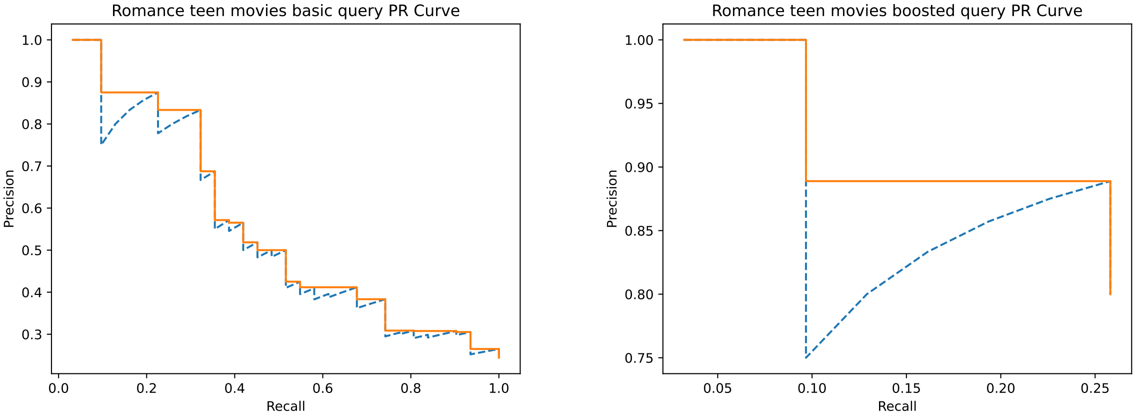


Figure 16. Romance Teen Movies Enhanced Schema PR Curves - Basic vs Boosted

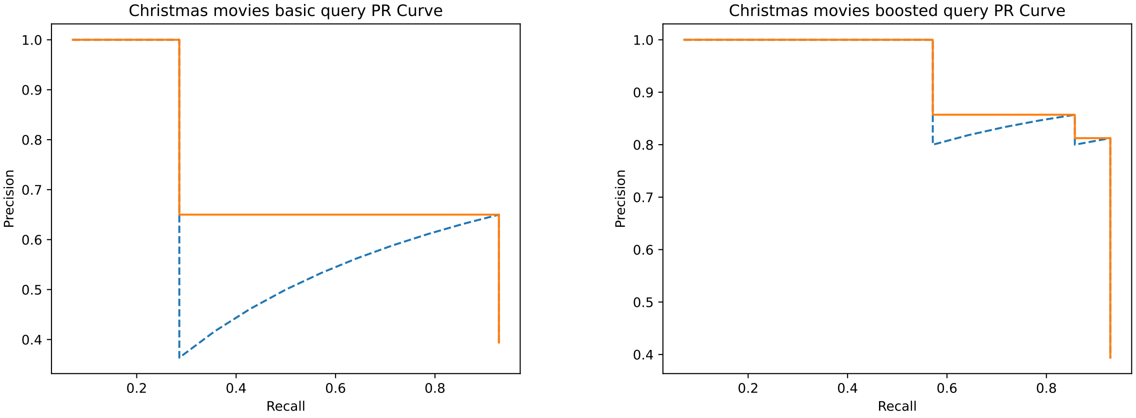


Figure 17. Christmas Movies PR Curves - Basic vs Boosted

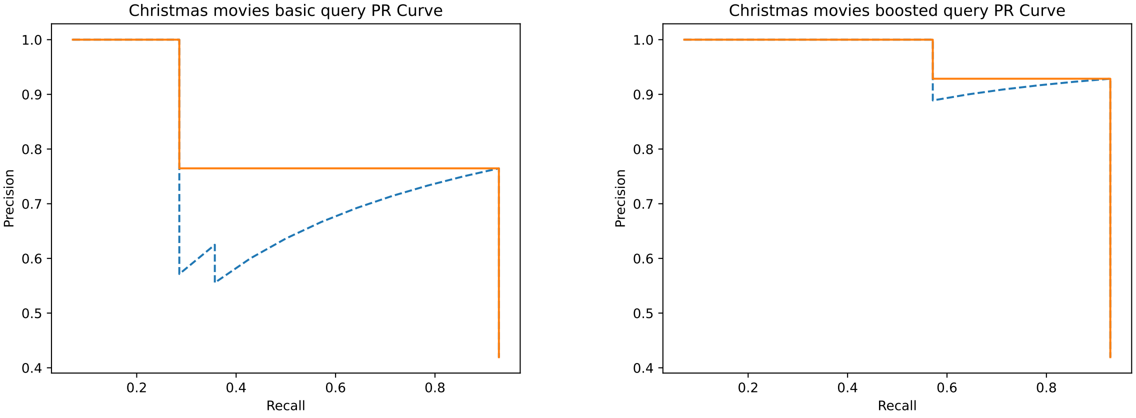


Figure 18. Christmas Movies Enhanced Schema PR Curves - Basic vs Boosted

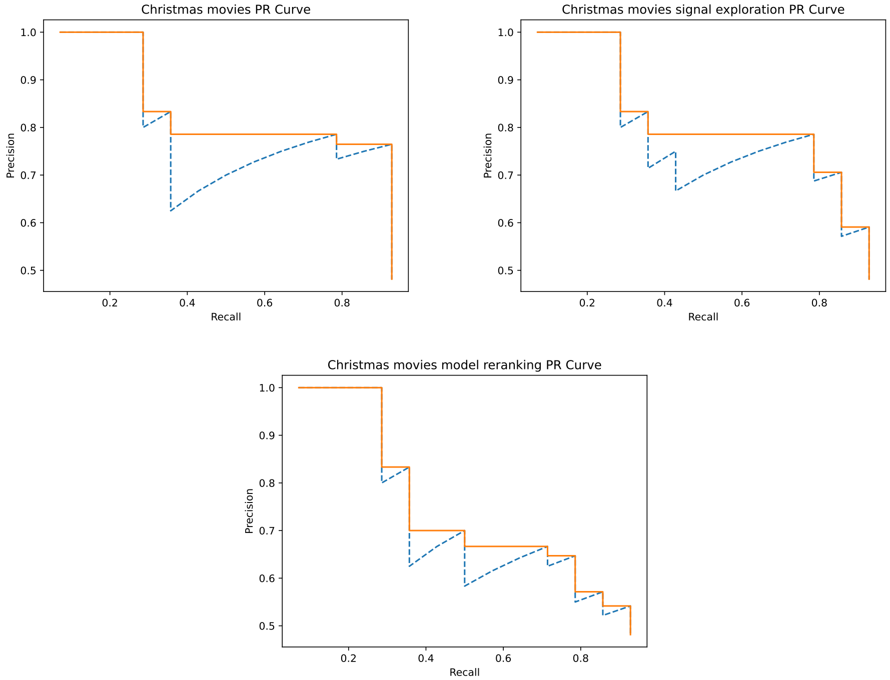


Figure 19. Christmas Movies hypotheses comparison PR Curves

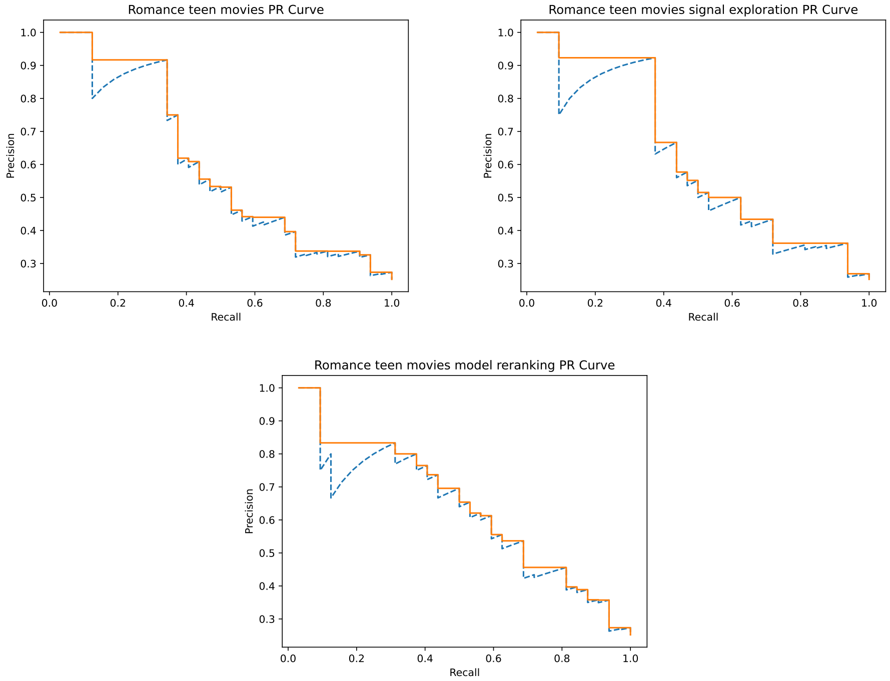


Figure 20. Romance Teen Movies hypotheses comparison PR Curves

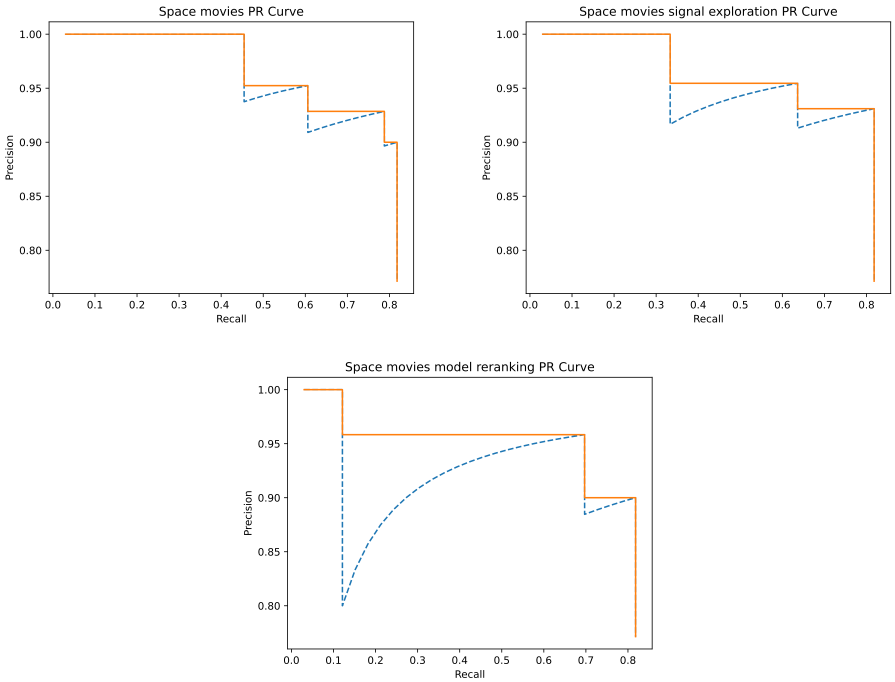
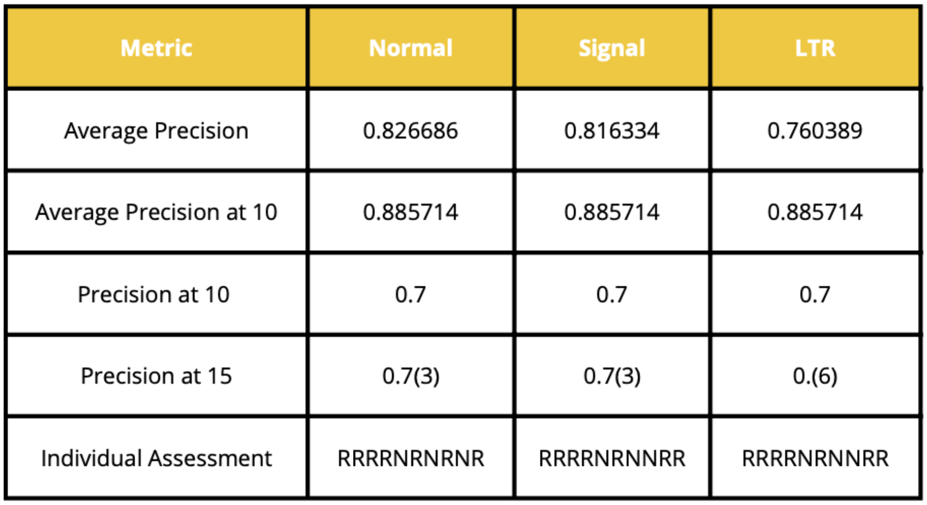
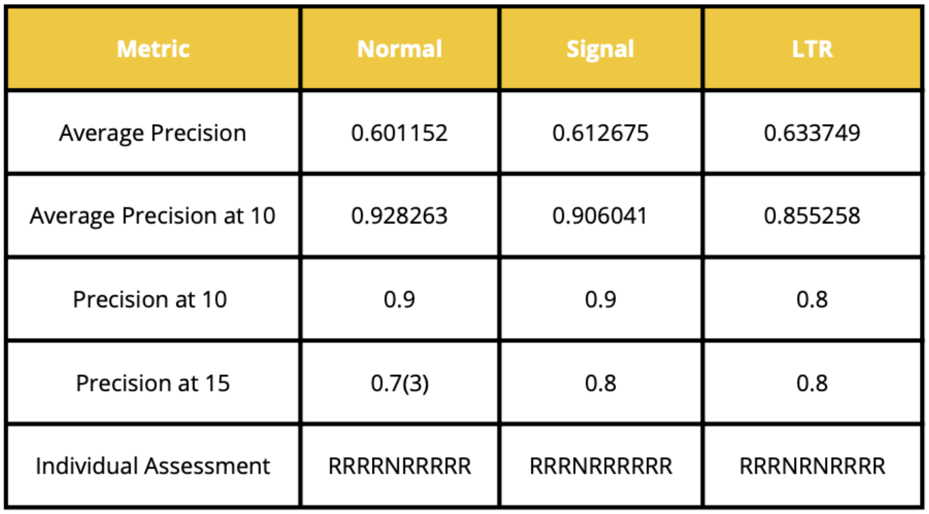


Figure 21. Space Movies hypotheses comparison PR Curves



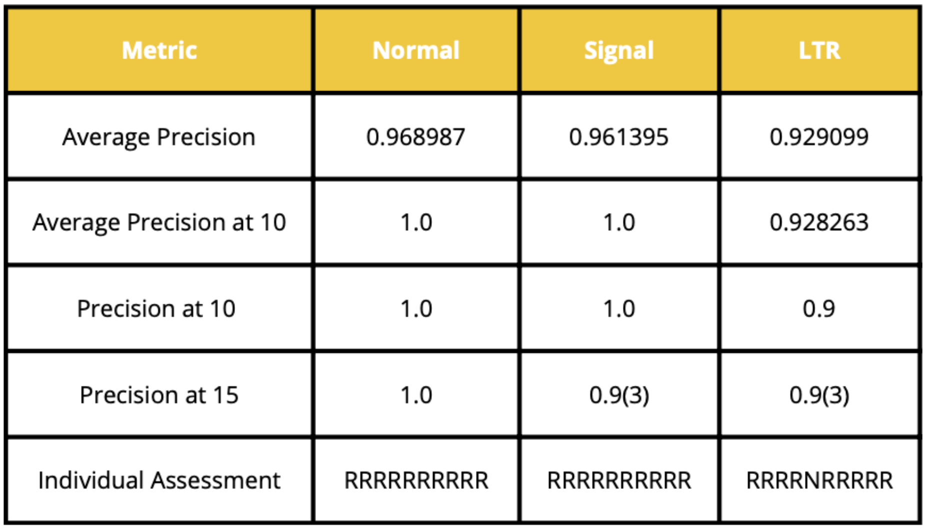
Metric	Normal	Signal	LTR
Average Precision	0.826686	0.816334	0.760389
Average Precision at 10	0.885714	0.885714	0.885714
Precision at 10	0.7	0.7	0.7
Precision at 15	0.7(3)	0.7(3)	0.(6)
Individual Assessment	RRRRNRNRNR	RRRRNRNNRR	RRRRNRNNRR

Figure 22. Christmas Movies metrics for hypotheses comparison



Metric	Normal	Signal	LTR
Average Precision	0.601152	0.612675	0.633749
Average Precision at 10	0.928263	0.906041	0.855258
Precision at 10	0.9	0.9	0.8
Precision at 15	0.7(3)	0.8	0.8
Individual Assessment	RRRRNRRRRR	RRRNRRRRRR	RRRNRNRRRR

Figure 23. Romance Teen Movies metrics for hypotheses comparison



Metric	Normal	Signal	LTR
Average Precision	0.968987	0.961395	0.929099
Average Precision at 10	1.0	1.0	0.928263
Precision at 10	1.0	1.0	0.9
Precision at 15	1.0	0.9(3)	0.9(3)
Individual Assessment	RRRRRRRRRR	RRRRRRRRRR	RRRRNRRRRR

Figure 24. Space Movies metrics for hypotheses comparison