FEUPFlix

Free-Text Movie Picker

Nuno Alves  
 Faculdade de Engenharia  
 Universidade do Porto  
 up202302778@fe.up.pt

João Carvalho  
 Faculdade de Engenharia  
 Universidade do Porto  
 up202004558@up.pt

Eduardo Silva  
 Faculdade de Engenharia  
 Universidade do Porto  
up202004999@up.pt

Adam Nogueira  
Faculdade de Engenharia  
 Universidade do Porto  
 up202007519@up.pt

ABSTRACT

In today's data-rich landscape, effective data processing and retrieval are paramount. This report documents the creation of a robust database and information processing and retrieval tool. Our project focuses on harnessing data related to movies to build a free search engine that assists users in selecting movies based on textual queries. We emphasize data quality, source reliability, and efficient retrieval mechanisms. This endeavor underscores the increasing importance of data management and information retrieval in modern society, particularly in the context of entertainment and movie selection.

**KEYWORDS**

*Data Processing, Data Retrieval, Information Retrieval Tool, Movie Search Engine, Project for Information Processing and Retrieval.*

1 INTRODUCTION

Our project is designed to address the challenges and opportunities presented by the vast landscape of movie-related data. We have selected the theme of movies due to the extensive historical data available in this domain, spanning decades of cinematic history. The movie industry continually accumulates data, making it a valuable area for information exploration and retrieval. Our main goal is to give the user the possibility to perform a free-text search in our tool and retrieve the response accordingly. i.e. the user can search “Horror Movies that are not Supernatural”, and the tool will provide the user movies that meet those requirements.

2 DATASET

In the pursuit of a rich and comprehensive foundation for our research, we delved into the realm of movie datasets, seeking a source that not only encapsulates a diverse array of films but also provides substantial information crucial to our objectives. This investigation led us through various datasets, each one with its own strengths, yet presenting unique challenges.

2.1 Dataset Choice

In the initial phase of our investigation, we explored various datasets related to movies, which contained a wealth of information, including details on personnel, ratings, reviews, and votes. This exploration strongly consisted in finding a good dataset, containing at least thousands of entries which we can preserve the maximum after data refinement. However, we encountered challenges with textual data, such as missing movie synopses or runtimes that were some of the fields that we consider more important to achieve our search goals, so we were forced to discard those datasets.

Ultimately, we opted for a Kaggle dataset [5] which, while smaller in scale (approximately 25000 movies), offered a much more complete data content. Although it still had some missing important fields, we decided to improve the quality of the dataset by merging it with additional data sources.

To enhance the dataset, we leveraged an API [6] to complete missing fields or correct malformed ones, thereby adding strength to the dataset.

2.2 Dataset Content

The original dataset that we've chosen for our project is composed of 12 columns, each containing valuable information crucial for both our research and the development of our movie search engine. These columns are curated as follows:

1. **Movie Title:** The title of the movie.
2. **Total Run Time:** The duration of the movie.
3. **Movie Rating:** The assigned rating for the movie.
4. **User Rating:** Ratings contributed by users for the movie.
5. **Genres:** The genres associated with the movie.
6. **Overview:** A concise summary of the movie.
7. **Movie's Plot Keywords:** Keywords pertaining to the movie's plot or theme.
8. **Director Name:** The name of the movie's director.
9. **Top 5 Cast Members:** The names of the top five cast members.
10. **Writer Name:** The name of the movie's writer.
11. **Releasing Year:** The year in which the movie was released.
12. **IMDb Movie URL Path:** The URL path leading to the IMDb page dedicated to the movie.

Following our data processing pipeline and the integration of data from "The Movie Database" (TMDb) API, we have refined our dataset to comprise the columns outlined below:

1. Movie Title
2. Genres
3. Overview
4. Keywords
5. Director
6. Top 5 Casts
7. Writer
8. Path

These refined columns not only provide a more comprehensive dataset for our movie search engine but also enhance the user experience by enabling more refined and precise queries. Our dataset's enrichment aligns with our project's goal of assisting users in selecting movies based on their textual preferences.

2.3 Dataset Quality and Source

Our dataset, originally obtained from IMDb.com, is now available on Kaggle. IMDb.com is a widely recognized and reputable platform for movie-related information, ensuring the reliability of our data source. The dataset's information was scraped directly from the IMDb public website, adding credibility to its content.

To enhance the dataset further, we have integrated the TMDB API [6] which allows us to improve data completeness and accuracy. Despite its origin as a Kaggle dataset, the core data's source remains IMDb.com, known for its trustworthiness in the realm of movie-related data.

Throughout our analysis, we found that the dataset met our criteria, containing the necessary information without any unexpected values or formats. This combination of a trusted source and quality assurance measures reaffirms the dataset's reliability and suitability for our research and movie search engine development.

3 PIPELINE

Our Data Preparation Pipeline is entirely constructed using Python scripts, where the pandas library played a crucial role in data management and manipulation. We harnessed the power of the requests library to facilitate communication between our Python scripts and the API, resulting in the creation of simple yet robust scripts for data cleaning and organization.

Our primary goal was to ensure the data's cleanliness and structure, ultimately formatting it into a CSV file for easy accessibility and further analysis.

3.1 Data Refinement

In our Data Refinement process, we followed a systematic approach within our pipeline:

1. **Handling Malformed Data:** The initial step involved the identification and removal of columns containing malformed or incomplete data from roman numeration in the year to prices in the runtime columns. To address this, we leveraged the API to correct the data, ensuring its accuracy and completeness.
2. **Eliminating Table Entries with Null Values:** Subsequently, we conducted a thorough examination of the dataset, removing any rows or entries with null values. This step helped us ensure data consistency and completeness removing a total of 2382 entries.
3. **Eliminating Duplicate Movie Titles:** We also identified and removed 651 movies with duplicated titles, ensuring that our dataset only retained unique and distinct entries.

By implementing these refinements, we enhanced the dataset's quality, removing irrelevant or erroneous information and streamlining it for our research and movie search engine development, getting a total of 21172 entries.

This means that after refinement, our final dataset remains 86,75% of the original dataset and with a strong improvement in the confidence of the data.

3.2 Data Analysis

In order to gain deeper insights into the dataset, we have developed a Python script featuring a range of functions aimed at extracting essential information. The knowledge we gather during the data analysis phase will play a pivotal role in guiding our decisions throughout the course of our project. As we delve into the visualizations and analyses presented below, we uncover valuable insights that will inform our project's direction and decisions. Our dataset, originally obtained from IMDb.com, is now available on Kaggle. IMDb.com Top of Form

A graph of a growing graph

Description automatically generated with medium confidence

**Figure 1:** Movies per Decade

The plot of the Figure 1 depicting the number of movies per decade reveals a trend in the cinematic landscape over time. The data showcases a gradual increase in the production of movies as we progress through the decades. This observation underscores the dynamic nature of the film industry, with each decade contributing more to the ever-expanding world of cinema.

As the plot unfolds, the most recent decade emerges as the most prolific in terms of movie production. This phenomenon reflects the contemporary cinematic landscape, where we often witness a surge in movie releases at the beginning of a new decade, as filmmakers and studios continue to explore new frontiers in storytelling and technology. The plot provides an intriguing glimpse into the evolving dynamics of the movie industry and sets the stage for further exploration and analysis in our project.

A pie chart with text on it

Description automatically generated

**Figure 2:** Movies Distribution per Genre

The Figure 2 plot offers a snapshot of the distribution of movie genres within our dataset. We've made the deliberate choice to exclude genres that represent less than 2% of the dataset to highlight the most prevalent categories.

As the plot reveals, the dominant genre is Drama, representing a portion at 22.9%. Action takes the second spot, accounting for 13.9% of the dataset. It's followed closely by Comedy with 13.2%, underscoring its popularity in the world of cinema.

This visualization allows us to quickly grasp the genre landscape within our dataset, providing valuable insights into the most common themes and styles that have captured the attention of moviegoers and filmmakers.

A graph with blue bars

Description automatically generated

**Figure 3:** Movies Rating Distribution

The plot shown in Figure 3 portraying the number of movies per rating paints a clear picture of the rating distribution within our dataset. A striking observation is that movies with a rating of 6 dominate the dataset, with more than 7000 instances. This rating level evidently encompasses a portion of the dataset, showcasing its prevalence among the films.

Conversely, the plot highlights that there are remarkably few movies with a perfect rating of 10. This scarcity of top-rated movies stands in contrast to the abundance of films clustered around the rating of 6, indicating the rarity and exceptional nature of movies that achieve a perfect 10.

This visualization guides our understanding of the dataset's rating distribution.

3.3 Pipeline Flow

A diagram of a software application

Description automatically generated**Figure 4:** Pipeline Flow

As we see in Figure 4, our data pipeline provides a comprehensive view of the journey our dataset has undertaken, from its initial source to its final refined state. The pipeline represents the orchestrated flow of data, incorporating several key stages, including data sourcing, cleaning, and refinement. It also showcases the integral role of the API in augmenting and updating the data for accuracy and completeness.

This pipeline underscores the significance of each step in the data processing journey, from handling malformed data to eliminating irrelevant entries and ensuring data consistency. It serves as a visual representation of the care and effort invested in curating a high-quality dataset for our research and movie search engine development.

By understanding this data flow, we can better appreciate the data's reliability, integrity, and suitability for our project's objectives. It also highlights our commitment to providing our users with a comprehensive and accurate resource for their movie-related inquiries.

3.4 Refined Data Structure

The final step in our data refinement process involved creating a structured CSV file containing all the refined data following a structured model as we can see in Figure 5.

In this structure we have some relations between the “Movie” and the other classes as we can see. We’ve forced each movie to have at least one genre, one and only one director, and 0 or more keywords, writers, and casts.

To accomplish this, we utilized a dedicated section of the Python script specifically designed for this task. The script was carefully crafted to include only the columns that met our research objectives and had undergone the necessary data refinement steps.

The Python script efficiently combined all these refined data elements and generated a CSV file, providing us with a clean and organized dataset in a structured format. This CSV file serves as a reliable foundation for our research and movie search engine development, enabling us to perform data analysis and deliver accurate and valuable results.

A screenshot of a computer

Description automatically generated

**Figure 5:** CSV Model

4 SEARCH GOALS

With all this data, we want to be able to take advantage of most of the attributes present. We want to give the user the ability of performing multiple movie search tasks, such as:

1. Search by genre name
2. Search by published year (can use ranges)
3. Search by title
4. Search by keywords

As a free-text search, the user can also combine multiple search parameters in one single query to better suit their needs.

It is important not only to allow a single category search at once but also combine them, for instance, if the user wants to watch an old romance movie it can simply query **“Romance movies before 1995”** as an example.

This shows us the large number of possibilities that our tool can perform if rightly used and we will cover this in the following sections.

The search tasks we decided to use are the following: “Movies to watch on Christmas eve”, “Teenage Romance movies”, “Horror movies that are not supernatural” and “Animated movies that feature princesses”.

5.1 JSON Format

Solr [1] can accept data in various formats, including CSV, which is the format in which we are storing the information. However, since Solr's default response format is JSON, and to facilitate easier data visualization and manipulation, we opted to convert our CSV data to JSON. This conversion is straightforward and enables us to work with Solr.

We developed a script to transform the data into JSON, resulting in a final file where each movie is represented as a JSON object composed of single key-value pairs, along with key-array values such as the "keywords" entry.

6 INDEXING DOCUMENTS

To start indexing and running search queries on our data, we first need to identify which fields are we going to run the search against. After that we can improve the user experience and the information retrieval performance.

6.1 Relevant Fields

To major improve our search results we decided that the most relevant fields to our tool will be “movie\_title”, “keywords” and “overview”. Those are the fields that we need to improve the Solr schema in order to get a fully functional free-text search engine.

6.2 Full-text Searches

In the context of full-text searches, we have chosen to index the "movie\_title", "keywords" and “overview” fields as they are central to searching for movies. Beginning with straightforward scenarios, these fields have been prioritized due to their significance in movie searches. In the Solr schema we create the “**overview\_keyword”** field type which is designed for full-text search on the “**overview”** and” **keywords”** fields. During indexing, the content undergoes tokenization, lowercase conversion, stop-word removal, and stemming. This enhances the search by making it case-insensitive, handling variations of words, and expanding the query to include synonyms. The synonym expansion is particularly useful for broadening the search scope, capturing related terms and improving the recall of the search. We also created the “**title”** field type is optimized for full-text search on the “**movie\_title”** field. Like “**overview\_keyword”**, it employs tokenization and lowercase conversion during both indexing and querying. This ensures that searches for movie titles are case-insensitive and can handle variations in input.

Tokens and Filters in the overview\_keyword Field Type:

1. Tokenizer - solr.StandardTokenizerFactory
   * This tokenizer breaks the text into words based on standard word boundaries. It is commonly used for general-purpose tokenization.
2. Filter - solr.ASCIIFoldingFilterFactory (preserving original):
   * This filter removes diacritics (accents) from characters, making the text ASCII-compatible. The option to preserve the original allows retaining the unaltered version of the text.
3. Filter - solr.LowerCaseFilterFactory:
   * Converts all characters in the text to lowercase. This ensures case-insensitive searching, treating uppercase and lowercase letters as equivalent.
4. Filter - solr.StopFilterFactory:
   * Removes common English stop words (e.g., "the," "and", "is") from the text. Stop words are often excluded from indexing to focus on more meaningful terms.
5. Filter - solr.PorterStemFilterFactory:
   * Applies the Porter stemming algorithm to reduce words to their root form. This helps in capturing variations of a word, improving search recall.
6. Filter - solr.SynonymGraphFilterFactory (with synonym expansion):
   * Expands the search by including synonyms of words present in the text. The synonyms are defined in an external file (**synonyms.txt**). This enhances search comprehensiveness.

Tokens and Filters in the title Field Type:

1. Tokenizer - solr.StandardTokenizerFactory:
   * Same as in the **overview\_keyword** field, this tokenizer breaks the text into words based on standard word boundaries.
2. Filter - solr.ASCIIFoldingFilterFactory (preserving original):
   * Removes diacritics and makes the text ASCII-compatible while preserving the original version.
3. Filter - solr.LowerCaseFilterFactory:
   * Converts all characters to lowercase for case-insensitive searching.

The **“overview\_keyword**” and “**title”** field types share common tokenizers and filters, ensuring consistent text processing during indexing and querying. The use of ASCII folding and lowercase conversion aids in normalizing the text, making it more conducive to effective searching. Stop word removal in both cases helps filter out common words that may not contribute significantly to the meaning of the text. The inclusion of synonym expansion in the **overview\_keyword** field broadens the search scope by considering synonyms defined in an external file.

These tokenizers and filters collectively contribute to a robust and flexible text analysis pipeline in SOLR, enhancing the search experience for the specified fields.

7 Queries

As previously discussed, the Solr query tool offers intriguing possibilities for exploration within our dataset domain. While there are numerous potential and relevant queries tailored to our domain, certain inquiries were left unexplored. This decision was based on their inability to meet our specific information requirements or their perceived limited impact on the average user's search tasks. In the following sections there will present some queries

7.1 Query 1- Christmas Eve

We are searching for movies suitable for Christmas Eve. The query aims to find titles related to "Christmas eve." The parameters for the query are designed to include movie titles containing "Christmas" and consider genres such as Comedy and Family, as well as specific keywords related to Christmas.

A screenshot of a computer

Description automatically generated

**Figure 6:** Parameters for query 1

7.2 Query 2 - Teen Romance

The query aims to find titles related to "teen romance." The parameters for the query are designed to include movie titles containing "romance" and consider the genres of Romance, along with specific keywords related to both "teen" and "romance."

A screenshot of a computer

Description automatically generated

**Figure 7:** Parameters for query 2

7.3 Query 3 - Horror Movie

We are looking for horror movies that are not supernatural. The objective is to identify titles associated with the horror genre while excluding those with a supernatural theme.

A screenshot of a computer

Description automatically generated

**Figure 8:** Parameters for query 3

7.4 Query 4 - Princess Animation

We are searching for animated movies that feature princesses. The aim is to identify titles that fall within the animation genre and include the theme of princesses.

A screenshot of a computer

Description automatically generated

**Figure 9:** Parameters for query 4

8 EVALUATION

Now that the system is operational and ready to carry out several searches, it's time to examine it and make the necessary adjustments to get the best results.

We will test two distinct situations to **examine** the system: schema without boosting, as well as schema with boosting and a few small query modifications. Precision and Recall will be used as the evaluation measures. Retrieval specificity, or the percentage of retrieved items that the user deems relevant, is measured by precision. This metric penalizes the system for retrieving irrelevant items, or false positives, but it does not penalize the system for failing to retrieve items that the user deems relevant, or false negatives. Recall penalizes false negatives but not false positives since it evaluates retrieval coverage, which is defined as the percentage of the set of relevant items that the system retrieves.

8.1 Experience 1 – Christmas Eve

This experience is related to query 2 mentioned in the section before.

|  |  |  |
| --- | --- | --- |
| **RANK** | **RELEVANCE NO BOOSTS** | **REVELANCE WITH BOOTS** |
| 1 | Yes | Yes |
| 2 | Yes | Yes |
| 3 | No | Yes |
| 4 | Yes | Yes |
| 5 | No | Yes |
| 6 | Yes | No |
| 7 | No | Yes |
| 8 | No | No |
| 9 | Yes | Yes |
| 10 | No | Yes |

**Table 1:** Movies retrieved without boosting and with boosting for Query 1.

A graph of a graph showing the difference between a number of objects

Description automatically generated with medium confidence

**Figure 10:** Precision Recall curves with/without boosting for Query 1.

|  |  |  |
| --- | --- | --- |
| **SCENARIO** | **METRIC** | **VALUE** |
| Schema without boost | Precision at 5 (P@5) | 0.6 |
| Average Precision | 0.79 |
| Schema with boost | Precision at 5 (P@5) | 1 |
| Average Precision | 0.93 |

**Table 2**: Precision results for Query 1.

Based on Table 1 and Figure 10, we can observe that the query performs better when we use the boosts because the results are much more relevant. Better precision results for the usage of queries with boosts are shown by the data in Table 2.

8.2 Experience 2 – Teen Romance

This experience is related to query 2 mentioned in the section before.

|  |  |  |
| --- | --- | --- |
| **RANK** | **RELEVANCE NO BOOSTS** | **REVELANCE WITH BOOTS** |
| 1 | Yes | Yes |
| 2 | Yes | Yes |
| 3 | Yes | Yes |
| 4 | Yes | Yes |
| 5 | No | No |
| 6 | No | No |
| 7 | No | Yes |
| 8 | Yes | Yes |
| 9 | No | Yes |
| 10 | Yes | No |

**Table 3:** Movies retrieved without boosting for Query 2.

A graph of a graph showing a blue and red line

Description automatically generated

**Figure 11:** Precision recall curve with/without boosting for Query 2.

|  |  |  |
| --- | --- | --- |
| **SCENARIO** | **METRIC** | **VALUE** |
| Schema without boost | Precision at 5 (P@5) | 0.8 |
| Average Precision | 0.87 |
| Schema with boost | Precision at 5 (P@5) | 0.8 |
| Average Precision | 0.89 |

**Table 4:** Precision results for Query 2.

From Table 4 and Figure 11 we can see that the boosts in this case made the results from the query a little bit more consistent. The precision results are also slightly better as we can see from Table 4.

8.3 Experience 3 – Horror Movie

This experience is related to query 3 mentioned in the section before.

|  |  |  |
| --- | --- | --- |
| **RANK** | **RELEVANCE NO BOOSTS** | **REVELANCE WITH BOOTS** |
| 1 | Yes | Yes |
| 2 | No | Yes |
| 3 | Yes | No |
| 4 | Yes | Yes |
| 5 | No | No |
| 6 | Yes | Yes |
| 7 | No | Yes |
| 8 | No | No |
| 9 | No | No |
| 10 | No | No |

**Table 5:** Movies retrieved without boosting and with boosting for Query 3.

A graph of a graph

Description automatically generated with medium confidence

**Figure 12:** Precision recall curve with/without boosting for Query 3.

|  |  |  |
| --- | --- | --- |
| **SCENARIO** | **METRIC** | **VALUE** |
| Schema without boost | Precision at 5 (P@5) | 0.6 |
| Average Precision | 0.773 |
| Schema with boost | Precision at 5 (P@5) | 0.6 |
| Average Precision | 0.83 |

**Table 6:** Precision results for Query 3.

In this encounter, the benefits of the schema and boosting are less obvious; Figure 12 illustrates this by showing how similar the two curves are. Nonetheless, given that the first two outcomes are more pertinent, we may observe that the boosting performs better initially before doing slightly worse later and somewhat better at the end. In table 4 we can also see that the boosting has a slightly better average precision in Table 6 which indicates better performance.

8.4 Experience 4 – Princess Animation

This experience is related to query 4 mentioned in the section before.

|  |  |  |
| --- | --- | --- |
| **RANK** | **RELEVANCE NO BOOSTS** | **RELEVANCE WITH BOOSTS** |
| 1 | Yes | Yes |
| 2 | Yes | Yes |
| 3 | Yes | Yes |
| 4 | Yes | Yes |
| 5 | Yes | Yes |
| 6 | Yes | Yes |
| 7 | No | Yes |
| 8 | Yes | Yes |
| 9 | No | Yes |
| 10 | Yes | Yes |

**Table 6:** Movies retrieved without boosting and with boosting for Query 4.

A graph with a blue line

Description automatically generated

**Figure 13:** Precision recall curve with/without boosting for Query 4.

|  |  |  |
| --- | --- | --- |
| **SCENARIO** | **METRIC** | **VALUE** |
| Schema without boost | Precision at 5 (P@5) | 1 |
| Average Precision | 0.96 |
| Schema with boost | Precision at 5 (P@5) | 1 |
| Average Precision | 1 |

**Table 7:** Precision results for Query 4.

Table 7 demonstrates that both situations produced positive outcomes for us; the first six results are relevant to both scenarios. It is better to use boosts than not to use any at all because doing so made the query's first ten results relevant. We can clearly say from the results that this query produced the best results for us.

8.5 Evaluation summary

The systematic evaluation of our information retrieval system across diverse scenarios revealed notable improvements with the incorporation of boosting strategies. These enhancements were particularly evident in precision-related metrics. The consistent high precision and favorable Average Precision values reinforce the overall success of the boosting strategies across different thematic contexts. These findings underscore the reliability and adaptability of the information retrieval system in delivering relevant and accurate results.

9 Search system improvements

To enhance our search system, we’ve selected a semantic search approach. Semantic search differs from traditional keyword-based search by incorporating the meaning of the query terms according to their context. It analyzes the context of words within the query and their relationships, aiming to comprehend user intent and leading to more relevant and accurate search results.

In addition to implementing a semantic search approach, we have also developed a user-friendly GUI interface for our search engine. This frontend serves as the interface through which users interact with our search system, making it more intuitive and accessible.

9.1 Semantic search

The use of semantic analysis through clever embeddings is an essential step in our quest to improve our movie search engine. Through an examination of crucial elements like "Overview," "Title," "Year," "Director," "Writer," "Genres," and "Keywords", we set out to uncover more in-depth information inside our movie collection.

9.2 GUI Interface

As mentioned, before we decided to implement a GUI interface, this interface will be using the semantic search with boosts previously talked about as well. In Figure 14 the search bar is displayed, prominently positioned on the main page, is the user's entry point to a vast library of films.

A screenshot of a computer

Description automatically generated

**Figure 14:** Search bar of GUI Interface.

Upon executing a search, the GUI instantly displays a concise list of the first ten results, offering users a quick overview of potential matches. Each result is presented with key information such as the movie title and overview which can be seen in Figure 15.

A screenshot of a computer

Description automatically generated

**Figure 15:** Results for query “serial killer”.

The option to explore more details about the results is then shown; if we choose it, a whole page with all the information taken from our dataset shows, as seen in Figure 16.

A screen shot of a movie

Description automatically generated

**Figure 16:** Page of movie “Chronicle of a Serial Killer”.

9.3 – Evaluation of semantic search.

To compare with the earlier findings displayed in section 8, we have chosen to evaluate the semantic search using the same queries previously investigated in part 7.

9.3.1 – Query 1 results using semantic search.

|  |  |
| --- | --- |
| **RANK** | **Semantic search with boosts** |
| 1 | No |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |
| 6 | Yes |
| 7 | No |
| 8 | Yes |
| 9 | No |
| 10 | Yes |

**Table 8:** Precision results for Query 1 using semantic search.

A graph with a line

Description automatically generated

**Figure 17:** Precision recall curve using semantic search with boosting for Query 1.

|  |  |
| --- | --- |
| **METRIC** | **VALUE** |
| Precision at 5 (P@5) | 0.8 |
| Average Precision | 0.71 |

**Table 9:** Precision results for Query 1 using semantic search.

As we can see from figures table 8 and table 1, and by analyzing the first 10 results of the semantic search, we can conclude that in this scenario using semantic search turn out to be not as effective as we would want to. The main reason of this occurrence is that the very first result of the query gave us a movie named “Holidays” which is a horror movie that we considered to not be a movie someone would like to watch on Christmas eve.

9.3.2 – Query 2 results using semantic search.

|  |  |
| --- | --- |
| **RANK** | **Semantic search with boosts** |
| 1 | Yes |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |
| 6 | No |
| 7 | Yes |
| 8 | Yes |
| 9 | Yes |
| 10 | Yes |

**Table 10:** Precision results for Query 2 using semantic search.

A graph with a line

Description automatically generated

**Figure 18:** Precision recall curve using semantic search with boosting for Query 2.

|  |  |
| --- | --- |
| **METRIC** | **VALUE** |
| Precision at 5 (P@5) | 1 |
| Average Precision | 0.95 |

**Table 11:** Precision results for Query 2 using semantic search.

By analyzing tables 10 and 3 we can confidently say that in this case using semantic search ended up improving a lot the results of this query. Only 1 non-relevant result which is major improvement compared to the best scenario we had previously which had 3 non-relevant results. The precision results shown in table 11 also indicate better performance.

9.3.3– Query 3 results using semantic search.

|  |  |
| --- | --- |
| **RANK** | **Semantic search with boosts** |
| 1 | Yes |
| 2 | No |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |
| 6 | No |
| 7 | Yes |
| 8 | Yes |
| 9 | No |
| 10 | No |

**Table 12:** Precision results for Query 3 using semantic search.

A graph showing a line

Description automatically generated **Figure 18:** Precision recall curve using semantic search with boosting for Query 3.

|  |  |
| --- | --- |
| **METRIC** | **VALUE** |
| Precision at 5 (P@5) | 0.8 |
| Average Precision | 0.77 |

**Table 13:** Precision results for Query 3 using semantic search.

Although there is less of an improvement in the use of semantic search in this query, when table 12 is analyzed and contrasted with the previous cases, as table 5 illustrates, this scenario produced more pertinent results. Figures 12 and 18's precision curves can be compared to see that the semantic search performs marginally better.

9.3.4 – Query 4 results using semantic search.

|  |  |
| --- | --- |
| **RANK** | **Semantic search with boosts** |
| 1 | Yes |
| 2 | Yes |
| 3 | Yes |
| 4 | Yes |
| 5 | Yes |
| 6 | Yes |
| 7 | Yes |
| 8 | Yes |
| 9 | Yes |
| 10 | Yes |

**Table 14:** Precision results for Query 4 using semantic search.

A graph of a graph showing a line

Description automatically generated

**Figure 19:** Precision recall curve using semantic search with boosting for Query 4.

|  |  |
| --- | --- |
| **METRIC** | **VALUE** |
| Precision at 5 (P@5) | 1 |
| Average Precision | 1 |

**Table 14:** Precision results for Query 3 using semantic search.

This query retrieved the ideal 10 results we expected, same as the previous scenario using boosts which can be seen in section 8.4.

9.4 Evaluation summary

Apart from edge-cases, the integration of semantic search has proven to be a significant improvement in our movie search engine. The enhanced precision, contextual understanding, and accommodation of natural language contribute to a more realistic and user-friendly search experience. Semantic search not only refines the search process but also adds substantial value by aligning more closely with users' natural language expressions, making the exploration of our movie database a more intuitive and rewarding endeavor.

CONCLUSION

As we delved into the dataset's content, examining key attributes such as movie titles, run times, ratings, user-contributed feedback, genres, plot summaries, keywords, and directorial information, we laid the groundwork for a multifaceted analysis. The fusion of data from diverse sources has fortified our dataset, addressed gaps and enhanced its overall quality.

The visualization of rating distribution has served as a guiding beacon, unveiling patterns that beckon further exploration.

As we embrace the Solr query tool, it’s intrinsic capability for boosting and metrics emerges as a pivotal asset. The ability to fine-tune and prioritize specific parameters through Solr boost enhances the precision of our queries, contributing significantly to the efficacy of our search tasks. Additionally, the design of the Solr schema supports the integrity and coherence of our dataset.

Looking forward, the confluence of our enriched dataset and the robust capabilities of Solr gives our project its own merit inside movie analytics, promising valuable insights and a well-structured and enhanced search experience for cinephiles and researchers alike.

REFERENCES

[1] Apache Solr Reference Guide, *https://solr.apache.org/guide/solr/latest/index.html*

[2] Sten Andler. 1979. Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY..

[5] DEY, U. (n.d.). *Kaggle*. Retrieved from Kaggle: https://www.kaggle.com/datasets/utsh0dey/25k-movie-dataset

[6] *TMDB*. (n.d.). Retrieved from TMDB: https://developer.themoviedb.org

A diagram of a software application

Description automatically generatedConference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00