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 meets 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 the 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 find 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 [[**https://www.kaggle.com/datasets/utsh0dey/25k-movie-dataset**](https://www.kaggle.com/datasets/utsh0dey/25k-movie-dataset)], 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.

In order to enhance the dataset, we leveraged an API from [[**https://developer.themoviedb.org/**](https://developer.themoviedb.org/)] 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 align 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 [<https://developer.themoviedb.org/>], 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 showed in the 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 generatedFigure 4: Pipeline Flow

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 the 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 the most of the attributes present. We want to give the ability to the user of performing multiple movie search tasks like:

1. Search by genre name
2. Search by publiched 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 his 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 show us the large amount of possibilities that our tool is able to perform if rightly used and we will cover this in the following sections.

5 SOLR

To build our information retrieval engine we’ve decided to use Solr as base tool to work on. We are running Solr 9.3 on a Docker container in order to orchestrate the work between us.

5.1 JSON Format

As we observed in the Solr documentation [1] Solr 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 firstly 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” and “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 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.

CONCLUSION

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..

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