Search A Movie An Information Retrieval Search Engine

Ioannis Tzortzakis
University of Antwerpen (Belgium)

***Abstract*—This paper is about an information retrieval system that assists people in finding movies easily on a website. You can search by the movie's title, description, or type of movie (genre). The website looks decent and works well, using HTML, CSS, and Javascript for the front-end, and Flask with the Whoosh library for the back-end. The system is tested, and it mostly gives the right information, even though sometimes the order is a bit unorthodox. To make it better, I suggest using machine learning to personalize the search results. Despite some small issues, the system is practical for finding movie information. This paper fits well with the publication's theme, making sure it's original, relevant, and easy for technical people to understand. All the publishing rules are respected and followed. The author hopes the process from submitting to publishing will be smooth.**

INTRODUCTION In today's digital world, there are a plethora of digital movies out there, making the role of information retrieval essential for navigating all the digital content. I created a special tool – a movie search system integrated into a web application. This paper is all about how I built it. The task of finding a movie to watch can be overwhelming with so many diverse options. My system makes this process easier for the user to find the movies he would have a preference, for by integrating his input and other elements from diverse sources. The main goal of this project was to apply the information retrieval elements I learned in class in a practical, real-life scenario of a project. You can find the implemented code in this repository https://github.com/JohnTzortz/IRProjectPython.

Digital Object Identifier 10.1109/MCE.YYYY.Doi Number
Date of publication 17 01 2024

Literature Review

The purpose of information retrieval is to swiftly and precisely acquire relevant information from a vast selection of information. Considering the vast and diverse data on the internet, information retrieval has become an extremely useful tool in the modern era of the internet.

The Internet as we know it now, has its base in the information retrieval service named WWW (World Wide Web). The Web gives users access to a vast array of mass media and content. Without the concept of IR (information retrieval), the Internet would not be possible due to the sheer volume of content and the massive growth of it these past years.

The most common information retrieval models are the boolean, Vector Space, and Probabilistic, which I'm utilizing in this search engine. I will not get into details about these models in this paper. While we have to recognize that these models were the basis of information retrieval, more modern search engines initialized advancements like machine learning and natural language processing, for more personalized information retrieval.

Structure of Project

The project could be divided into three main parts.

- *Front end*—The interface. What the user can see.
- *Back end*—The inner workings. The searching algorithm and the processing of the dataset.
- *dataset*—A set of information (data) used to retrieve information.

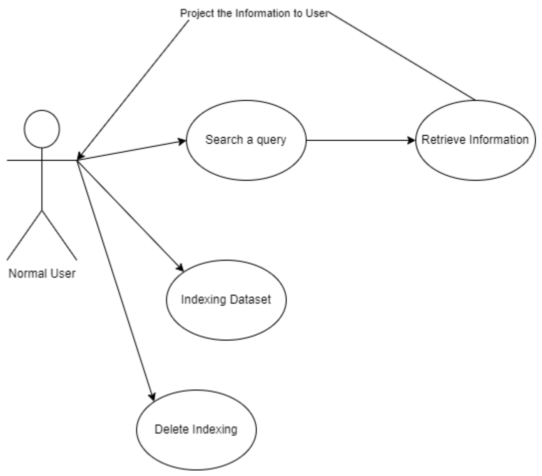


Figure 1. Use Case Diagram of Project: Search A Movie.

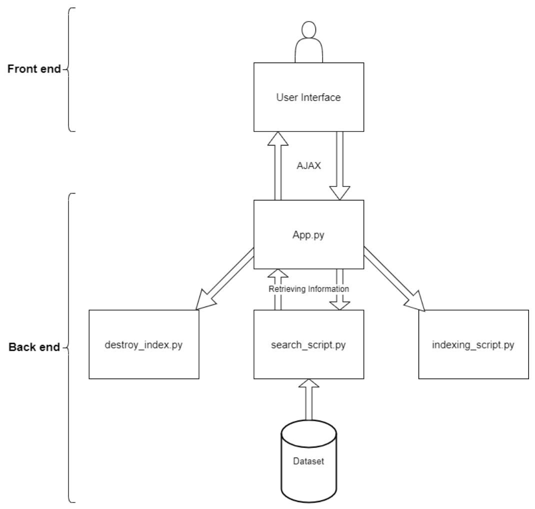


Figure 2. Main Structure of Project.

This method, splitting the project into distinct separate ”blocks” is very common and allows for specialization, modularity, scalability, collaboration, flexibility, and security of the project. This approach is called ”Divide and Conquer” and be found in many aspects of computer science as well as other aspects of life [1].

Front-end

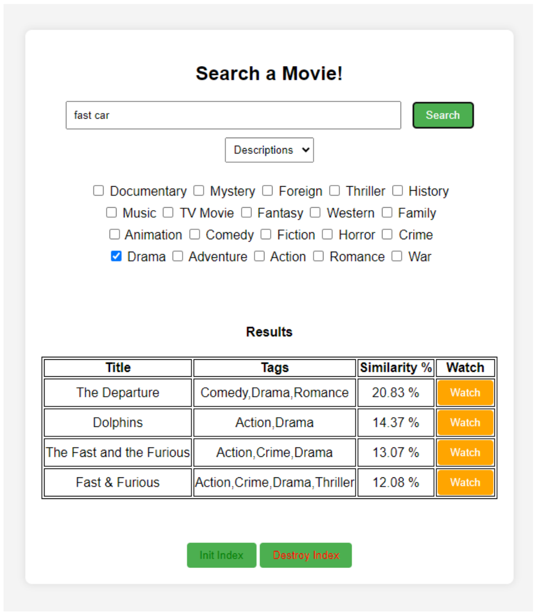
The project has an easy-to-use interface. It is quite familiar to an internet user because its core design is very similar to the biggest search engine, Google[2].

The technologies used to construct the interface are HTML, CSS, and Javascript. The communication between the Front and Back-end is achieved with AJAX (Asynchronous JavaScript And XML). The technologies I used are very common in web development, although the use of them is through frameworks; contrary to what I did. The reason I didn’t make use of frameworks is purely to keep the interface simple and channel the focus to the back-end/Information retrieval part of the project.

There is a search bar in the middle of the application window, where the user can type the phrase/word he wants the retrieved movies to be related to. Next to the bar, there is the search button. Under the bar, there is a dropdown menu where the user can choose based on what characteristic will the search be performed. Under the search bar are the filters that represent different genres that will filter the results before are shown.

The retrieved results are projected in an HTML table. This is not so common nowadays. Usually, other applications like Netflix, projects the results in carousels making them more presentable for the user. Still, for the need of this project, I decided that a table could keep it simple and efficient. The table has four cells per row. The first one is the title of the movie. The second is the genre of that movie. The third is the percent of matching score between the movie and the term the user searched. The last one is a simple button that is not doing something for the moment but could be used to improve the search algorithm. More on this later on.

Under all of these, are two buttons used to manage the indexing, a process that I will refer to later on. As the names of these buttons indicate, they create and destroy the indexes according to their names. The role of these buttons is to make the indexing processes abstract from the user.


Search a Movie!

fast car

Descriptions

☐ Documentary ☐ Mystery ☐ Foreign ☐ Thriller ☐ History
☐ Music ☐ TV Movie ☐ Fantasy ☐ Western ☐ Family
☐ Animation ☐ Comedy ☐ Fiction ☐ Horror ☐ Crime
☑ Drama ☐ Adventure ☐ Action ☐ Romance ☐ War

Results

Title	Tags	Similarity %	Watch
The Departure	Comedy,Drama,Romance	20.83 %	Watch
Dolphins	Action,Drama	14.37 %	Watch
The Fast and the Furious	Action,Crime,Drama	13.07 %	Watch
Fast & Furious	Action,Crime,Drama,Thriller	12.08 %	Watch

Figure 3. The interface (front-end). The term ’fast car’ is searched

Table 1. Example of searching the query ”fast car” in descriptions with the filter ”Drama” being pressed

Name of Movies	% Score
The Departure	20.83%
Dolphins	14.37%
The Fast and the Furious	13.07%
Fast & Furious	12.08%

These are the results as shown in the graphic interface.

Back-end

The main structure of the backend is a Flask application. I chose Flask for the backend because it’s simple and easy to use, making it great for building web apps. Flask is known for being straightforward and flexible, which fits well with my project’s focus on information retrieval. It helps keep the backend structure efficient.

Python has lots of tools for working with data, which was a big factor in my decision. Python’s flexibility and the many libraries available made it a smart choice for handling the information retrieval parts of the project. Using Python just made sense because it has all the tools I needed for processing and retrieving data.

For the information retrieval part, I made use of the library Whoosh[3]. I used the library to access the dataset and make the indexing. Indexing is the process of organizing and structuring data to facilitate efficient and quick retrieval of information. It involves creating a roadmap or catalog that allows for faster access to specific data points within a larger dataset. Whoosh makes this process abstract by using Python functions. Whoosh library is utilizing the Okapi BM25F ranking function[4]

I programmed the searching algorithm that makes use of the multiParsers Whoosh provides to retrieve the most related information from the dataset. This algorithm initially receives a maximum of 5% of the initial dataset which later on is filtered and returns the top 15 results. The reason behind the 5% is purely for performance issues. The percentage is possible to be adjusted accordingly to the size of the dataset.

Although whoosh is doing some light preprocessing on the query, tokenizing the query is not enough. Therefore is not possible to search more than a phrase.

Dataset

The dataset I used is ”The Movies Dataset” from Kaggle after my professor recommended it to me. It’s a good dataset with movie scores, ideal for training machine learning models. Even though it had more info than I needed, I did some preprocessing to make fit my project better. Kaggle datasets are trusted, so it’s a solid foundation for my work. More information about Keggle datasets can be found on the official website https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv

Getting rid of common words (stop words) during indexing makes searches more accurate. Words like ”and” and ”the” don’t add much meaning. In information retrieval, these words can cause confusion and less precise results. Skipping them during indexing helps the search focus on important words, making results more relevant and accurate. It simplifies the process, concentrating on key terms, and boosts the overall performance of the search engine.

Other tools

Other tools were used to make this project. There was heavy use of ChatGPT[5] due to a lack of experience working with Python and Whoosh.

Evaluation

In the evaluation of the project, it's important to emphasize the role of the BM25F ranking function employed by the Whoosh library. This ranking model significantly contributes to the accuracy of search results. Unlike simpler models, BM25F takes into account factors such as term frequency and document length normalization, making it well-suited for real-world datasets. The advanced nature of this ranking model allows it to assign precise scores to documents based on their relevance to a given query, resulting in more refined and accurate search outcomes. The implementation of BM25F[4] in the Whoosh library enhances the overall effectiveness of the search engine, ensuring that the retrieved documents closely align with the user's search intent.

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)},$$

Figure 4. BM25F ranking function.

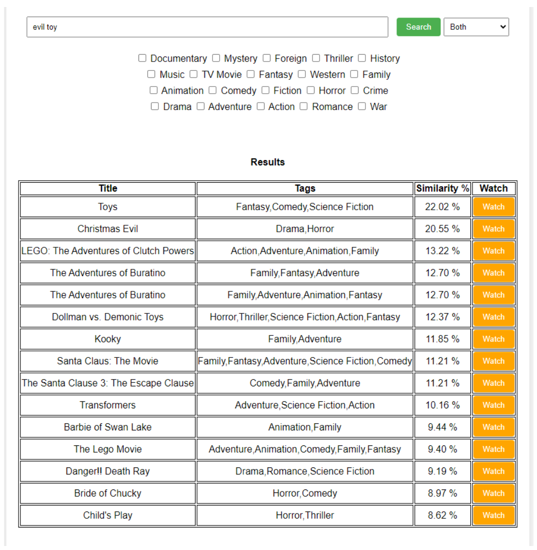
Where:

- D represents a document,
- Q represents a query,
- q_i represents the i-th term in the query,
- n is the number of documents containing the term q_i,
- N is the total number of documents,
- $|D|$ is the length of document D (number of terms),
- avg_dl is the average document length,
- k_1 is a positive tuning parameter,
- b is another tuning parameter (usually set between 0.5 and 0.8).

Here is a demo video of the program working. Keep in mind the user interface is from a previous version that didn't include the scores in the table. Excluding this information, the rest of the processes (setup, searching the query, etc) are the same as the video https://www.youtube.com/watch?v=ihbKl4dz_Lk. In the code files, you can find a readme.txt file that has all the necessary information to set up and run the program.

Whoosh library returns a score for each result it retrieves, which indicates how matching is that information to the query. This method of evaluating retrieved information is commonly called "relevance ranking" or "retrieval ranking".

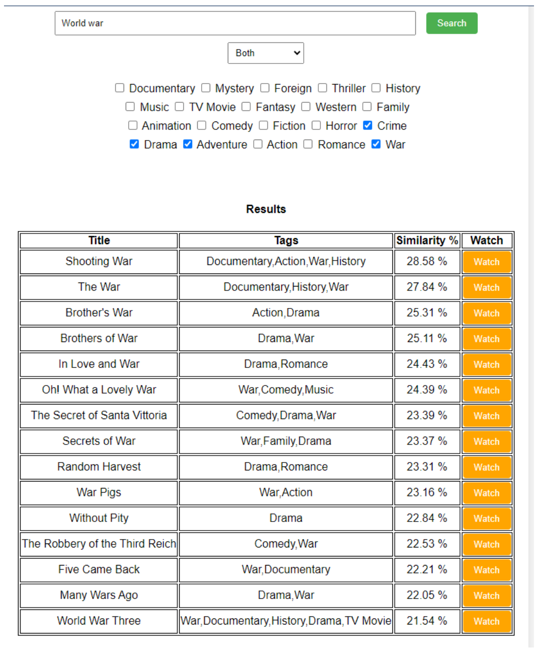
Many times upon inspecting the information retrieved, the user would wait to find some movies ranked higher than what they are. This is simply not the case here because the algorithm is performed purely based on the ranking model. Modern search engines consider other factors, like personalized searches and trends, etc.


evil toy
Search
Both
Documentary Mystery Foreign Thriller History
Music TV Movie Fantasy Western Family
Animation Comedy Fiction Horror Crime
Drama Adventure Action Romance War

Results

Title	Tags	Similarity %	Watch
Toys	Fantasy,Comedy,Science Fiction	22.02 %	Watch
Christmas Evil	Drama,Horror	20.55 %	Watch
LEGO: The Adventures of Clutch Powers	Action,Adventure,Animation,Family	13.22 %	Watch
The Adventures of Buratino	Family,Fantasy,Adventure	12.70 %	Watch
The Adventures of Buratino	Family,Adventure,Animation,Fantasy	12.70 %	Watch
Dollman vs. Demonic Toys	Horror,Thriller,Science Fiction,Action,Fantasy	12.37 %	Watch
Kooky	Family,Adventure	11.85 %	Watch
Santa Claus: The Movie	Family,Fantasy,Adventure,Science Fiction,Comedy	11.21 %	Watch
The Santa Clause 3: The Escape Clause	Comedy,Family,Adventure	11.21 %	Watch
Transformers	Adventure,Science Fiction,Action	10.16 %	Watch
Barbie of Swan Lake	Animation,Family	9.44 %	Watch
The Lego Movie	Adventure,Animation,Comedy,Family,Fantasy	9.40 %	Watch
Danger!! Death Ray	Drama,Romance,Science Fiction	9.19 %	Watch
Bride of Chucky	Horror,Comedy	8.97 %	Watch
Child's Play	Horror,Thriller	8.62 %	Watch

Figure 5. Example 1.


World war
Search
Both
Documentary Mystery Foreign Thriller History
Music TV Movie Fantasy Western Family
Animation Comedy Fiction Horror [✓ Crime]
[✓ Drama] [✓ Adventure] Action Romance [✓ War]

Results

Title	Tags	Similarity %	Watch
Shooting War	Documentary,Action,War,History	28.58 %	Watch
The War	Documentary,History,War	27.84 %	Watch
Brother's War	Action,Drama	25.31 %	Watch
Brothers of War	Drama,War	25.11 %	Watch
In Love and War	Drama,Romance	24.43 %	Watch
Oh! What a Lovely War	War,Comedy,Music	24.39 %	Watch
The Secret of Santa Vittoria	Comedy,Drama,War	23.39 %	Watch
Secrets of War	War,Family,Drama	23.37 %	Watch
Random Harvest	Drama,Romance	23.31 %	Watch
War Pigs	War,Action	23.16 %	Watch
Without Pity	Drama	22.84 %	Watch
The Robbery of the Third Reich	Comedy,War	22.53 %	Watch
Five Came Back	War,Documentary	22.21 %	Watch
Many Wars Ago	Drama,War	22.05 %	Watch
World War Three	War,Documentary,History,Drama,TV Movie	21.54 %	Watch

Figure 6. Example 2.

Discussion

Results

The results of the engine are what I was anticipating. The scores are correct and in the correct order.

The main differences with other mainstream movie search engines are the lack of personalized results and the volume of the retrieved information. The lack of personalized results can be fixed when the engines get integrated with machine learning, a topic I'll expand on later on. Although the volume of retrieved information depends on the size of the dataset, the way data is projected in the interface plays a huge role in not to overwhelm the user, something my project is not prepared to do efficiently.

Challenges in the Coding Process

Creating this project posed challenges, affecting the choice of programming language and the coding environment. Initially, attempting to code in Java using the Lucene library turned out to be quite demanding. The lack of reliable online resources made the learning process difficult, and the complexities of Java Lucene resulted in setbacks. This led to a reassessment of the chosen technology.

Shifting to Python seemed promising, and the Whoosh library emerged as a suitable alternative for information retrieval. However, transitioning to Python had its challenges. Limited access to the command line in the university's coding environment made installing external libraries problematic. To overcome this, the project adapted by using the packages within the integrated development environment (IDE). This adjustment facilitated the successful integration of the Whoosh library, highlighting the importance of flexibility in overcoming unforeseen challenges.

This experience emphasizes the need to choose technologies aligned with project goals and available resources. It also underscores the value of resilience and adaptability when facing obstacles in the dynamic field of software development.

Proposed Enchancements

Searching algorithm

As mentioned before, most modern search algorithms utilize machine learning to provide a more accurate search for the results to be in line with what the user might expect when searching. At the moment the project is not expanding to machine learning but there were some proposals for upgrading.

There could be a login system; therefore a cookie system that will collect data about the user so the experience could get personalized for each user. The watch button could also collect data from the user and train the algorithm to show the type of movies the user chooses more often.

After watching a movie, the user could provide feedback about the movie itself. This could make a general ranking of movies so the algorithm will fetch better quality movies.

User Experience

Upon starting the program, occasionally the screen freezes and a page refresh is necessary. This could be fixed by upgrading to better frameworks that have already been used. Also, optimizing the program's installation processes will upgrade the user's experience,

Concluding Sections

Appendices

In the files of the project, there are instructions and other information about the installation of the project. If a problem occurs while installation, please contact me (info at the end).

Acknowledgment

I want to express my sincere thanks to Professor Toon Calders for his helpful slides that greatly aided my understanding. I'm grateful to Mr. Ewoenam Kwaku Tokpo, our TA, for his insightful support during the project. I also made extensive use of ChatGPT, which proved instrumental in accelerating my Python coding skills.

CONCLUSION

In summary, this project successfully created a movie search engine within a user-friendly Flask web application. By using Whoosh for information retrieval, we built a system that works well and responds efficiently. While we met our main goals, there's room for improvement in the future, especially by adding machine learning to enhance result ranking. The lessons and skills gained from this project have been valuable for my development in information retrieval and web development.

Looking ahead, the core of this project, especially the backend infrastructure, holds potential for broader applications. While the current implementation may not serve as a standalone API, adapting it for other projects is a feasible endeavor. This highlights the scalability and adaptability of the developed backend, opening doors for integration into various information retrieval systems. The insights gained and skills developed in this project contribute to future explorations in refined applications for upcoming projects.

REFERENCES

1. Smith, D. R. (1985). The design of divide and conquer algorithms. Elsevier B.V.
2. Search Engine Market Share Worldwide — StatCounter Global Stats. StatCounter Global Stats. Archived from the original on December 10, 2020. Retrieved April 9, 2021
3. Information Retrieval Library for Python. Official documentation: https://whoosh.readthedocs.io/en/latest/intro.html
4. Stephen E. Robertson, Hugo Zaragoza (2009) - The Probabilistic Relevance Framework: BM25 and Beyond, Foundations, and Trends® in Information Retrieval 3(4):333-389
5. ChatGPT, personal communication, February 11, 2023, https://chat.openai.com/

Ioannis Tzortzakis University of Antwerpen, Antwerpen, Belgium. Contact him at ioannis.tzortzakis@student.uantwerpen.be