Master en sciences et technologies de l'information et de la communication

INFO-H410 - Techniques of AI

Content-based Movie Recommendation System

Souha Belhaj Rhouma

Aya Iftissen

Salma Namouri

Prof. Hugues Bersini

Academic year 2024-2025

Table des matières

1 Introduction

Abstract

In this project, we present a content-based movie recommendation system that suggests films similar to a user-selected title by analyzing their textual metadata. The data used is from The Movie Database (TMDB) [The25]. The system extracts a number of factors (features) such as genres, cast, crew, keywords, and plot overviews. These features are combined into a unified textual representation then vectoriezed. Using cosine similarity, we compute the similarity between movies which enables the system to return the top 5 most similar movies for the selected input. We implemented a user friendly interface built using Streamlit [Str25] to present the final product. The system shows the effectiveness of content-based filtering in generating relevant and intuitive movie suggestions.

Introduction

Nowadays, we use recommendation systems everywhere ; while shopping online, watching a movie, listening to music or even when watching reels on Instagram. It's a system that understands our behavior and based on it, suggests products or services. These systems guide us through a vast information space in those modern online platforms. This project focuses on implementing a content-based recommendation system that recommends movies similar to a user-selected title based on textual and categorical metadata.

Unlike collaborative filtering approach that rely on user behavior and preferences, content-based systems utilize the inherent attributes of items themselves. By analyzing movie features like the title, genre, keywords, crew, this system learns the descriptive profile of each movie and proposes others that have in common characteristics. The purpose of this project is not only to provide meaningful movie recommendations but also to demonstrate how content-based approaches can be effectively applied when user data is unavailable or limited.

2 Recommendation System Approaches

Recommendation systems are algorithms designed to suggest relevant items to users based on various data sources. Over the years, many approaches of filtering have emerged. This chapter will serve as a research on the three main approaches of recommendation systems ; collaborative, content-based, knowledge-based. In this project, we use the content-based recommendation approach.

2.1 Collaborative-based Filtering

This system is based on the assumption that users who shared similar preferences in the past are likely to prefer similar items in the future. It only relies on past user behavior (e.g. previous product ratings) and relationships between users and inter-dependencies among products as shown in Figure 2.1. This means that there exists 2 types of collaborative filtering ; user-based nearest neighbor recommendation, item-based nearest neighbor recommendation.

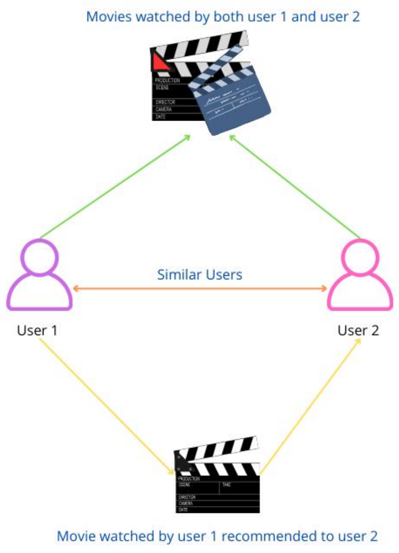

Movies watched by both user 1 and user 2
Similar Users
User 1
User 2
Movie watched by user 1 recommended to user 2

Figure 2.1 : Illustration of Collaborative-based Recommendation System.

2.1.1 User-based nearest neighbor recommendation

In this approach, the objective is to identify users who have a similar preference pattern to the active user (the one for whom we want to make a recommendation). We will first represent each user as a vector of interactions with items. Then, we compute the similarity between the active user and all other users using a measure of similarity such as cosine similarity. After that, we select the top K similar users (the nearest neighbors making it similar to KNN [1]). And finally, we predict the interest level of the active user for a given item by aggregating the ratings from their neighbors.

2.1.2 Item-based nearest neighbor recommendation

The focus shifts from users to items. Rather than finding similar users, the algorithm represents each item as a vector of user ratings. Then it computes similarity between items using measures like cosine similarity. After that, for a target item, it identifies the most relevant and similar items. Finally, it predicts how a user would rate the target item based on how they rated similar items in the past.

2.1.3 Strengths

It offers various advantages. One of the biggest strengths of collaborative filtering is that it is independent of the content or features of items. The system does not require any domain-specific knowledge, metadata, or item descriptions. This makes it highly adaptable in many fields. In addition, collaborative filtering has the ability to capture trends, behaviors, and preferences from the entire user community, not just an individual's profile.

2.1.4 Limits and challenges

The most common challenge is the huge data volumes since we have millions, if not billions, of users and items. Added to that, this approach suffers from the cold start problem which means that it requires a lot of user interaction and data to get accurate outputs. It can also be very confusing when finding a missing value, as it could mean either the value is really missing or something more important. This missing value isn't always a random empty element of the data, it could be useful information, for example, the user doesn't like the movie so they don't rate it or don't even interact with it.

2.2 Knowledge-based Filtering

This approach uses explicit domain knowledge, business rules, or user-provided constraints to generate recommendations. It's often used when there's no purchase history and when the preferences are specific. Instead, they use explicit knowledge about users and items to generate recommendations based on logical rules [LZW21]. This approach is driven by structured item description like price, size, location, ...etc.

2.2.1 Strengths

This approach solve some problems faced by the previous recommendation system. Since knowledge-based recommenders do not depend on past user behavior, they work well in cold-start situations. They also provide transparent reasoning behind each recommendation, often citing specific criteria or constraints. Lastly, it gives the opportunity for users to explicitly define what they want which leads to a precise recommendation aligned with their current needs (good for marketing and promoting).

1. A ML algorithm that helps in identifying the K nearest data points.

2.2.2 Limits

Despite all the advantages it offers, knowledge-based recommendation systems have their drawbacks as well. Unlike ML-based models, these recommenders are less adaptive since they do not improve automatically over time through user feedback or implicit learning. Added to that, recommendations are based on strict logical rules or constraints resulting in overly narrow output.

2.3 Content-based Filtering

This type of filtering suggests items to a user based on the attributes of items the user has shown interest in. In other words, It relies on the availability of description of items and of a user profile that gives importance to such items. This type of system is widely used in recommendation systems that focuses on the intrinsic characteristics of items and individual user preferences. The illustration in Figure 2.2 simply explains the concept.

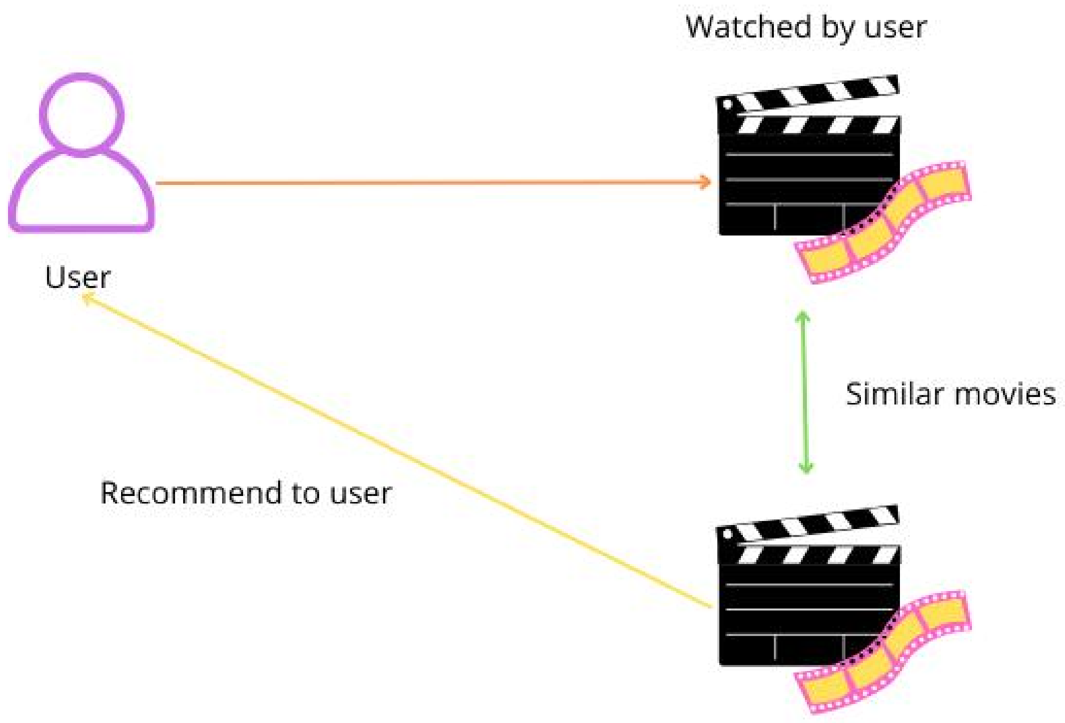


Figure 2.2 : Illustration of Content-based Recommendation System

2.3.1 How it works

These recommenders rely on item metadata, such as genre, keywords, descriptions, tags or specifications. They build a user profile based on the content of the items the user has previously liked or interacted with. Recommendations are then generated by finding new items with similar content.

2.3.2 Strengths

One of the most prominent advantages is that content-based filtering is user independent which means that it does not require data from other users. It operates entirely based on the user’s preferences and features of the item selected. This makes it extremely powerful in terms of privacy-sensitive contexts. Moreover, on the item side, it is effective when dealing with new or unpopular items. as long as the item provides descriptive metadata, it can be recommended, even if no other user has interacted with it. This solves the cold start issue. It’s also easier to fine-tune the recommendation process by selecting important features, applying weights, or filtering undesirable attributes making this recommendation approach more controllable and flexible.

2.3.3 Limits

The system recommends items similar to what the user already knows resulting in a lack of diversity and limited exploration. The cold start problem is solved on the item side but it’s still an issue on the user side. When the new user has not interacted with any item, the system lacks the basis to generate recommendations.

3 Methodology

Here we describe the core components and processing pipeline of the content-based movie recommendation system. The system is designed to recommend similar movies by analyzing textual attributes associated with each movie and computing their semantic similarity.

3.1 Data Collection & Preprocessing

3.1.1 Data Collection

We got the data from TMDB 5000 movies dataset [The17] downloaded from Kaggle. The TMDB 5000 Movie Dataset includes metadata for over 5k movies sourced from The Movie Database (TMDb) [The25]. It contains information like cast, genres, release dates, budget, revenue, and user ratings. It is widely used for building and evaluating movie recommendation systems and for data analysis projects in film and entertainment analytics. In this dataset, we have two tables (csv files), credits and movies that will be concatenated together.

3.1.2 Data Preprocessing

In this step, we clean and transform the raw movies data to prepare it for vectorization and similarity analysis. First, we created tags which is a merge of relevant textual columns. This combined text provides a summarized description of each movie, capturing its key characteristics in a single field for easier comparison and analysis.

After this we normalize the data to reduce the noise. This normalization happens as follows, we remove punctuation and whitespace and lowercased the text to prevent inconsistency during tokenization. Next, comes stemming, that plays an important role in preparing the textual data used to represent each movie. Raw text contains variations of the same word that may reduce the effectiveness of comparison. For instance, the tags of one movie might contain the word *performer* while another has *performance* or *performing*. Although these words are from the same lexical field, the computer treats them as distinct words.

To tackle this issue, we used PorterStemmer from the NLTK library to perform **stemming**. This refers to the process of reducing words to their root. This normalization helps in making the cosine similarity calculations more accurate.

3.2 Feature Extraction

In feature extraction, it is essential to convert movie descriptions into a structured, numerical form. This technique is crucial as it ensures that semantically similar words are treated as identical, which improves the consistency and efficiency of text matching. We applied the stemming function for each row of the dataset to stem the tokens in the tags column. This helped us minimize redundancy in the textual data and enhanced the system’s ability to identify content-based similarities between movies. This stemmed and cleaned text served as the foundation of vectorization using CountVectorizer which convert the textual data into numerical vectors. More specifically, we limited the vocabulary to the 5000 most frequent words while excluding common English stopwords [Cou23]. This vectorized representation allowed us to quantify movie similarity based on word occurrence and formed the basis for the cosine similarity calculation.

3.3 Similarity Computation

Once feature vectors were generated, we compute the similarities between movies using cosine similarity metric. In Figure 3.1 we illustrate how cosine similarity works. Cosine similarity evaluates the cosine of the angle between two vectors in a multi-dimensional space. If two movies have very similar tags, based on word context and frequency, their corresponding vectors will point in nearly the same direction, resulting in a cosine similarity score close to 1. Otherwise, the similarity score approaches 0.After the calculation is done, a square similarity matrix is produced. Each entry [i,j] in this matrix represents how similar two movies are. This matrix is then stored in a pickle file for optimization purposes using Python’s pickle module which avoids recomputing the similarity matrix on every app load.

When referencing to the table at runtime, the system directly retrieves the top N most similar movies to a given input, enabling fast and accurate content-based recommendations.

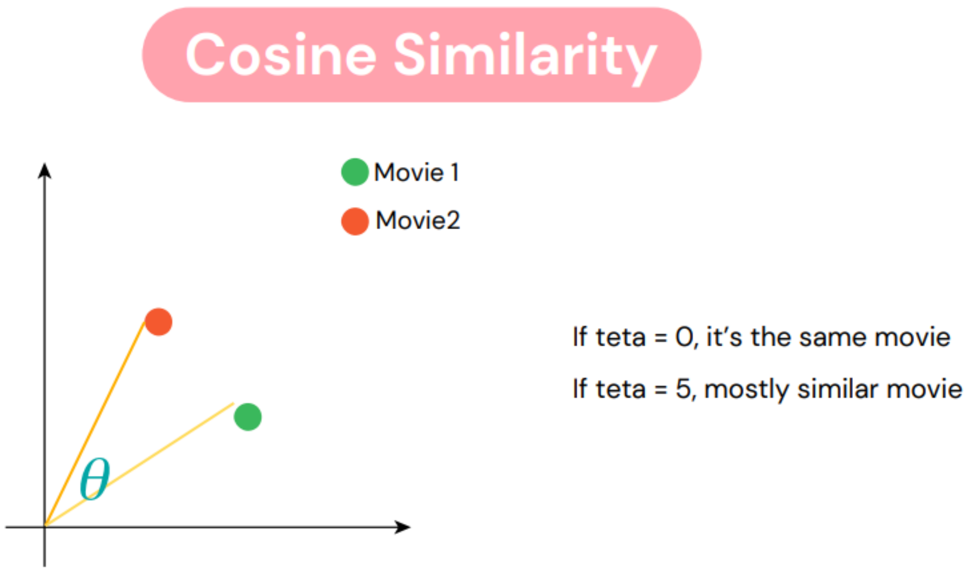

Cosine Similarity
Movie 1
Movie2
θ
If teta = 0, it’s the same movie
If teta = 5, mostly similar movie

Figure 3.1 : Simple Illustration of Cosine Similarity

4 Recommendation Algorithm

The recommendation engine is the core component of this project. It uses a precomputed similarity matrix to rapidly identify and retrieve movies with the most similar content. The algorithm operates in real time, providing users with fast and relevant suggestions with minimal latency. This algorithm has 4 core steps that we will walk through in this chapter.

4.1 Movie Index Retrieval

Starting off with the retrieval of movie index, when the user selects a movie, the system first retrieves the index of the selected movie from the pre-processed dataset. This index corresponds to the row and column of the cosine similarity matrix where the similarity scores for that movie are stored.

4.2 Similarity Score Extraction

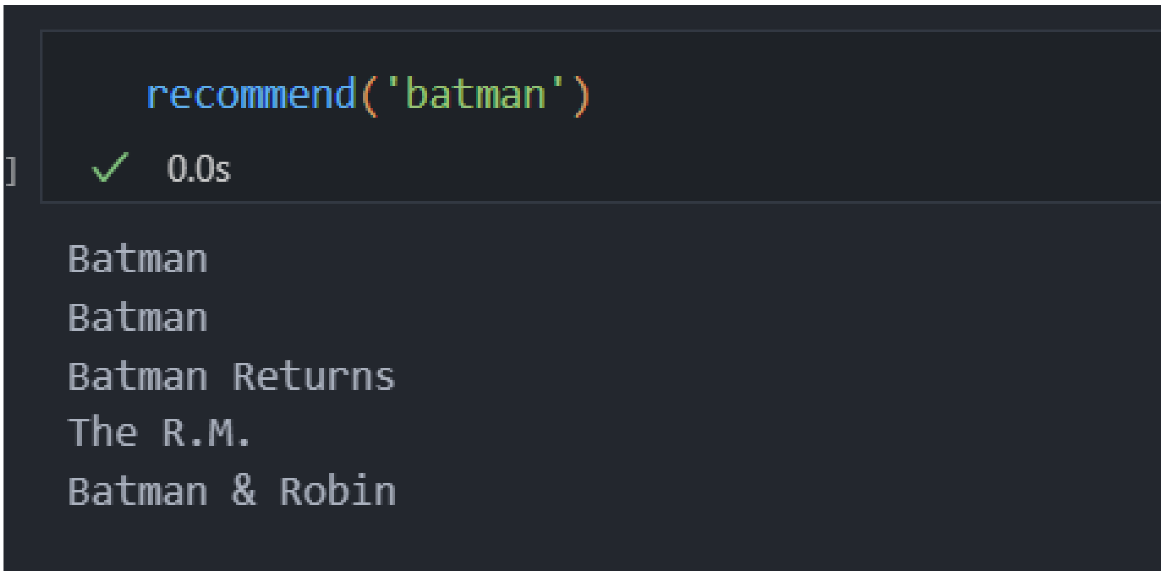
Using the index obtained from the previous section, the system extracts the score of similarity between selected movie and others from the dataset. We do this by accessing the corresponding row in the cosine similarity matrix. Each entry in this list is a tuple of the form (movie_index, similarity_score), where similarity_score reflects how similar the two movies are based on their content vectors.

4.3 Ranking Similar Movies

To find the most relevant recommendation, the system classifies the list of similarity scores in descending order based on the similarity score values ensuring that the movies with the highest semantic resemblance to the selected one appear first. The first element of the list is going to be the selected movie itself, with a similarity score of 1.0 since it's identical to itself (same movie), so it is explicitly excluded from the final recommendations.

4.4 Top-N Recommendation Selection

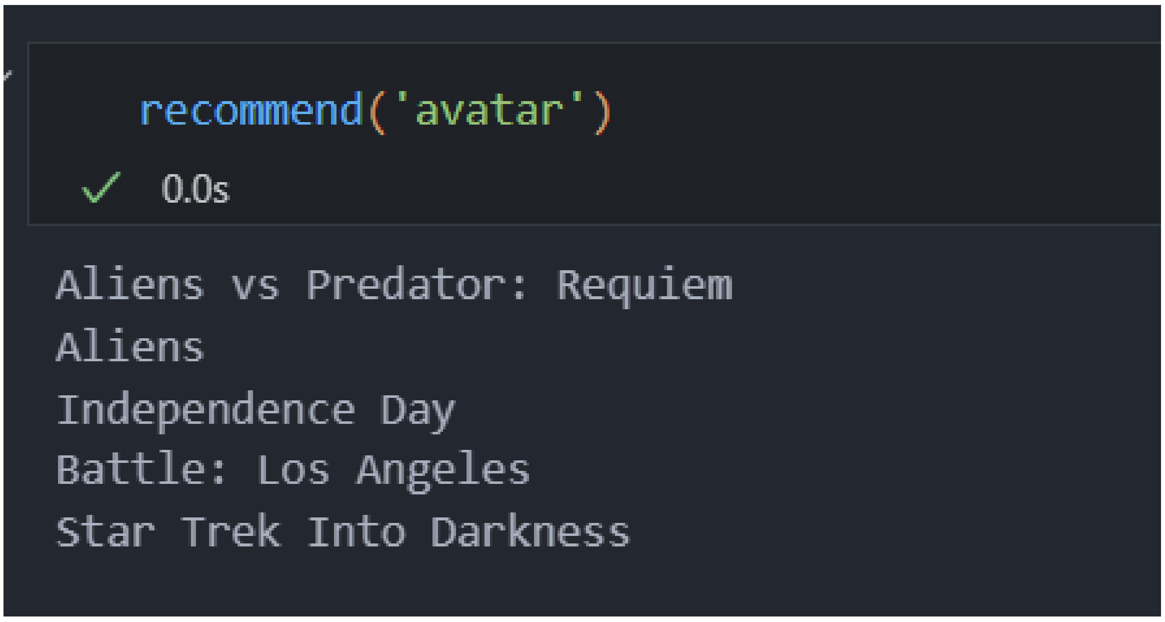
Lastly, the system selects the top 5 most similar movies, skipping the user's selected movie. This serves as the final recommendations returned to the user. The Figures 4.1 and 4.2 below show and example of the output we get.



```
recommend('batman')
✓ 0.0s

Batman
Batman
Batman Returns
The R.M.
Batman & Robin
```

Figure 4.1 : Example 1 of the recommended movies



```
recommend('avatar')
✓ 0.0s

Aliens vs Predator: Requiem
Aliens
Independence Day
Battle: Los Angeles
Star Trek Into Darkness
```

Figure 4.2 : Example 2 of the recommended movies

4.5 User Interface

To provide users with a simple and visually appealing way to interact with the recommendation system, we developed a web-based user interface using Streamlit, a Python framework designed for rapid development of interactive data applications.

For a better user experience, we retrieve the movie name and the poster images of these recommended movies using the TMDB API and displayed in the Streamlit interface created as shown in Figure 4.3 and **??**.

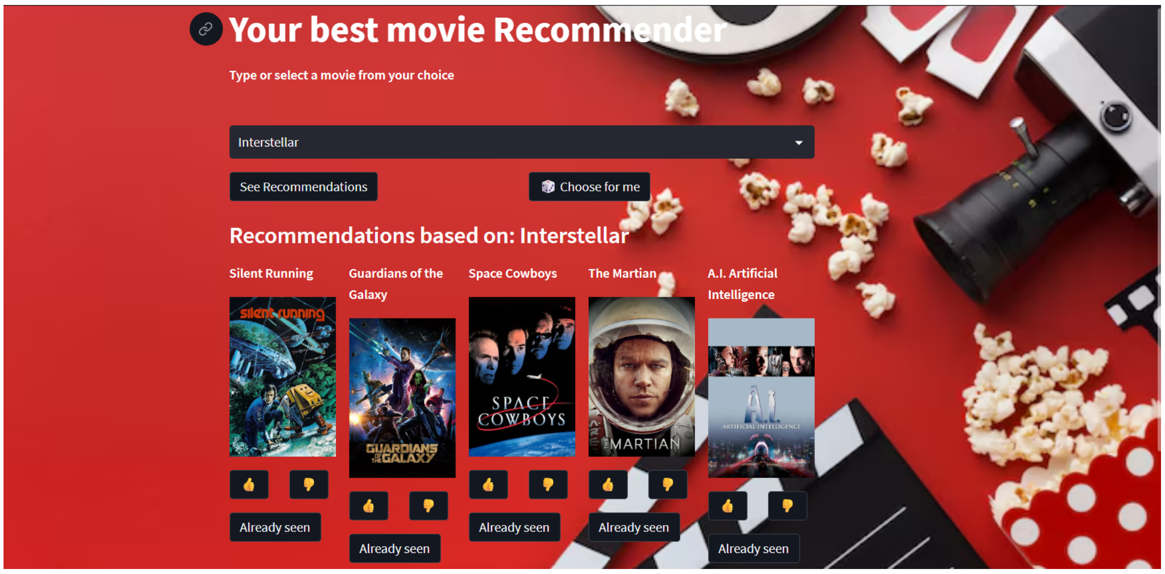


Figure 4.3 : Screenshot 1 of the interface of the content-based recommendation system

The technical flow goes as follows ; first the user will choose a movie and clicks on the button ”See Recommendations”. This interaction triggers the recommender engine, which uses precomputed similarity metrics to suggest five related movies. The system uses lightweight user feedback in the form of ”like” and ”dislike” buttons. Disliked movies are stored in session memory and automatically excluded from future recommendation batches through rule-based filtering. While this does not involve any model updates or learning, it creates a more personalized experience by adapting visible suggestions in real time. If the dislike button was pressed, then the movie gets replaced with another movie.

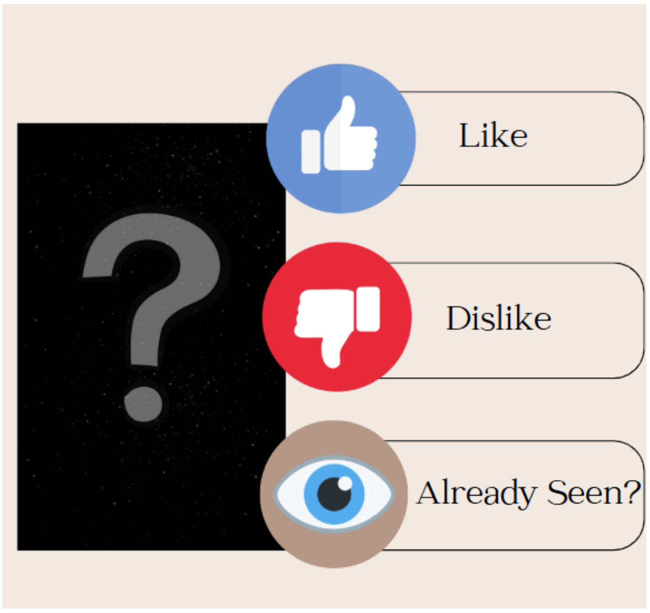


Figure 4.4 : Options to help the content-based recommendation

One of the feature is the ”Choose for me” button, which enhances user engagement through surprise-based exploration. Instead of waiting for explicit user input, this functionality samples a base movie from the user’s watched history and suggests an unseen, non-disliked movie. This introduces a form of randomized exploration, showing suggestions without requiring manual input every time.

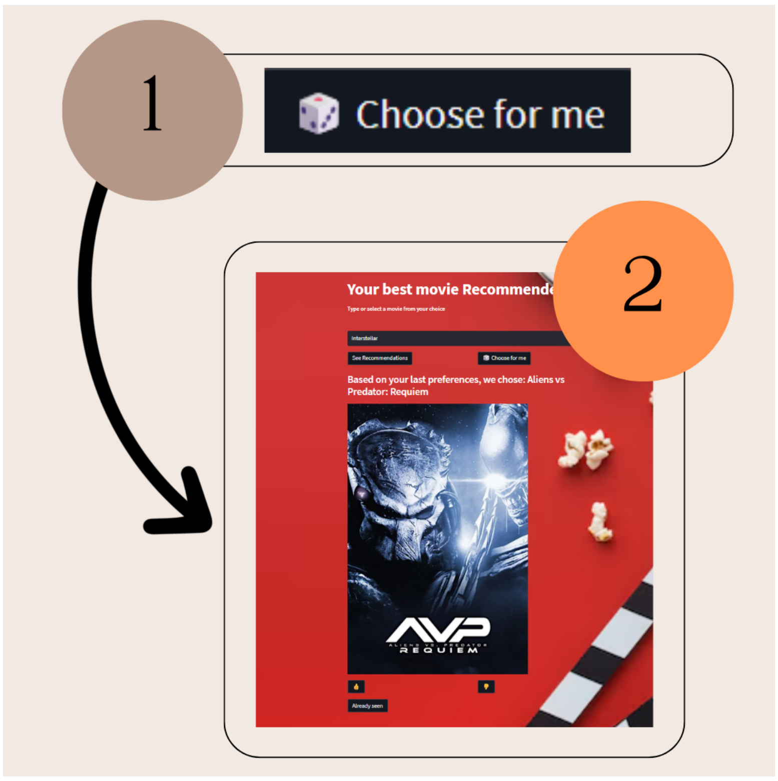

1
Choose for me
2
Your best movie Recommend
Type or select a movie from your choice
Interstellar
See Recommendations
Choose for me
Based on your last preferences, we chose: Aliens vs Predator: Requiem
AVP REQUIEM
Already seen

Figure 4.5 : When choose me is triggerred, it'll show the best recommendation based on all your preferences

The interface maintains state across interactions using Streamlit's session state. This includes the user's liked and disliked movies, previously shown recommendations, and current UI content. This memory-like behavior allows the application to avoid showing the same content multiple times and ensures continuity as the user continues to explore more movies.

Implementation Note

For running the app.py, one of us had an issue with the pushed version. So if you get the errors do the following steps :
- Try changing the line containing the path to the saved file with adding an extra t to the dataset in both line 64 and line 65. This will make the path similar to the path where the precomputed similarity matrix was saved.
- Another thing that was an issue, if you press the dislike or like button and get an error, try changing all lines with this st.experimental_rerun() to this : st.rerun()

Again, this is just in case you faced an error when running the streamlit but it works fine for all of the group members.

5 Results and Conclusion

5.1 Results

The content-based movie recommendation system performs well and efficiently in suggesting movies that are thematically similar to the user's input. These recommendations are based on content features such as genre, keywords, and crew, all of which were embedded into the system through pre-processing and vectorization. We provided in the previous chapters some screenshots of the results we got. Here's another screenshot that recommends movies similar to one of the movies in the dataset.

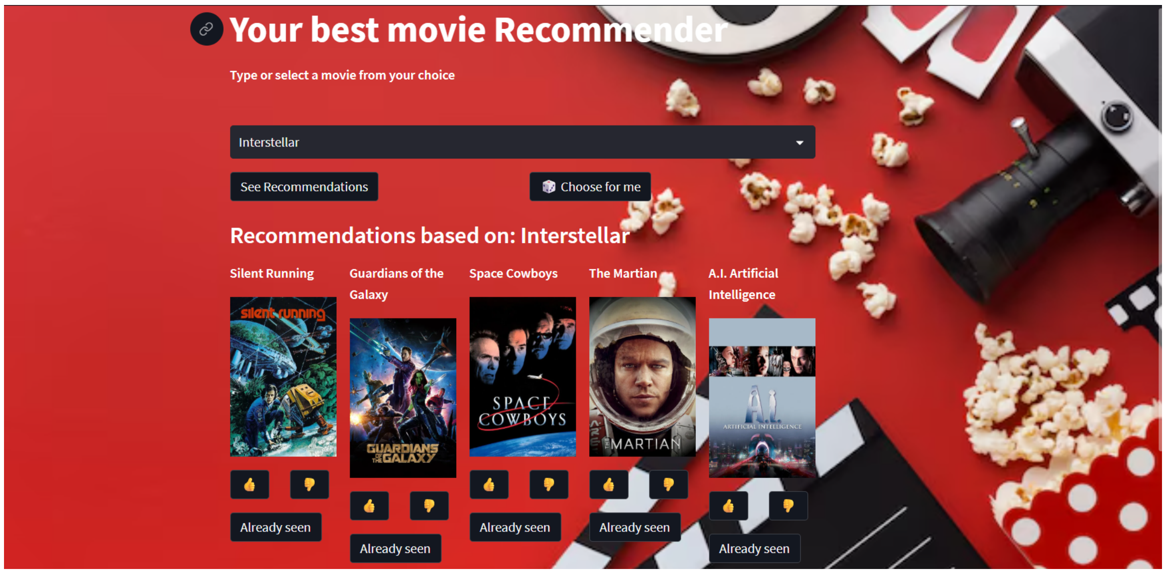


Figure 5.1 : Screenshot of the interface result to recommend movies

5.2 Conclusion

This project successfully demonstrates a content-based movie recommendation system that leverages natural language processing techniques and basic machine learning principles. By transforming textual metadata into structured numerical vectors and applying cosine similarity, the system mimics how a model might learn semantic relationships between items. Although it does not involve supervised learning or training on labeled data, it still showcases the core idea of feature extraction and similarity-based decision-making. This reflects an important category of machine learning applications focused on unsupervised or heuristic-based methods. Furthermore, it provides a solid foundation for scaling toward more advanced systems, such as hybrid recommenders or deep learning-based models.

5.3 Future work

One promising direction for future development is integrating the like and dislike buttons into a feedback loop. These user preferences could serve as implicit rewards in a reinforcement learning framework. Over time, the system could learn to personalize recommendations more effectively by updating its policy based on observed user behavior, improving the relevance of suggestions with continued interaction.

Bibliographie

[Cou23] Coursera (2023). *What Are Stop Words ?* Accessed : 2025-05-20. url : https://www.coursera.org/articles/what-are-stop-words.

[LZW21] Li, X., Y. Zhang et Z. Wang (2021). « Movie recommendation system using collaborative filtering ». In : *Information Sciences* 578, p. 241-257. doi : 10.1016/j.ins.2021.07.005. url : https://www.sciencedirect.com/science/article/abs/pii/S0020025520307568.

[Str25] Streamlit (2025). *Streamlit : A faster way to build and share data apps*. Accessed : 2025-05-20. url : https://streamlit.io/.

[The25] The Movie Database (2025). *The Movie Database (TMDb)*. Accessed : 2025-05-20. url : https://www.themoviedb.org/.

[The17] The Movie Database (TMDb) (2017). *TMDB 5000 Movie Dataset*. Accessed : 2025-05-20. url : https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata.