Movie Recommendation System

Prof. Sanober Shaikh[1], Adhithyaram S[2], Aryak Deshpande[3]

[1]Professor, Dept. of Information Technology, Thadomal Shahani Engineering College, Maharashtra, India
[2]Student, Dept. of Information Technology, Thadomal Shahani Engineering College, Maharashtra, India
[3]Student, Dept. of Information Technology, Thadomal Shahani Engineering College, Maharashtra, India

---***---

Abstract – *Recommendation systems have been around for a while now with the advent of websites for movies, books, products, music, etc. There are various techniques used in the core of a recommender engine but the end-user does not have any say in it. In this movie recommender system, we have implemented two such techniques which are collaborative filtering and content-based filtering. The users can choose between the two and customize the parameters affecting the recommendations according to their preferences.*

Key Words: **Recommender System, Machine Learning, Collaborative Filtering, Content based Filtering**

1. INTRODUCTION

Recommender systems are have become a staple in this era of the internet economy. They help in reducing the overload of information by providing customized information access. Modeling, programming, and deploying these recommender systems have allowed businesses to enhance revenues and retain customers. These systems include customized search engines, personalized shopping agents, and handcrafted content indices. The scope of personalization and use of these systems extends to many different areas, not just web pages. The algorithm and concepts used in recommender systems can range from keyword matching in user profiles, content-based filtering, collaborative filtering, to more sophisticated techniques of data mining such as clustering server logs. Recommendation systems filter data using various concepts and approaches and recommend the most relevant items to users based on customizable criteria. It first captures the past behaviour of a customer and recommends products based on that. This proves to be beneficial to the user and clients as users only see relevant content and are not bombarded with unnecessary products and clients get better engagement. Hence it has become imperative for businesses nowadays to build smart recommendation systems and make use of the past behavior of their users.

2. DOMAIN OVERVIEW

2.1 Content-based filtering

The content-based filtering approach tries to study the liking of a user given the movies' features, which the user reacts positively to. Content refers to the features or attributes of the movies the user likes. The gist of this technique of recommendation is to compare movies using specific attributes, understand what the user has already liked, predicting what he may like, and recommend a few movies from the database with the most similar attributes. These attributes can be fixed or specified by the user himself. This method of recommendation uses movie features to recommend other similar movies to what the user has already liked.

This technique involves the calculation of the cosine similarity matrix which is done using the movie's feature vectors and the user's preferred feature vectors from the user's previous records. Then, the top few movies which are most similar are recommended to the user.

User's feedback and its significance in recommendation:

Implicit Feedback: The user's likes are recorded based on actions like clicks, searches, etc.

Explicit Feedback: The users specify their liking by actions like reacting to an item, marking it as their favorite, or rating it. In our system, the user marks certain movies as 'favorites' and also gives ratings to movies. The ratings given are used for collaborative filtering.

Cosine Similarity:
It is a concept used to measure how similar one movie is to every other movie in the dataset. It measures the cosine of the angle between two vectors projected in a multi-dimensional space.[6] Even if the two similar movies are far apart by means of the Euclidean distance, they may still be oriented closer together. [5]

$$Cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\ \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a}\cdot\vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

Fig. 2.1: Cosine similarity for content-based filtering

Using Python, we can easily calculate the count matrix using count vectorizer and its method. From the count matrix, the cosine matrix can be evaluated.[8]

2.2 Collaborative Filtering

Collaborative filtering is commonly used for data that cannot easily be described by attributes from the dataset such as books and movies. First, the prediction technique builds a user-item matrix of preferences for items by users. It then matches users by computing the similarities between their data and makes recommendations.

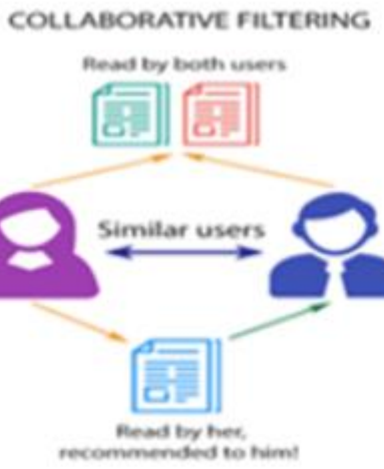


Fig. 2.2: Collaborative Filtering[7]

The following example helps us understand this concept better. If person X likes the 3 movies - Interstellar, Shutter Island, Extinction, and person Y likes Shutter Island, Extinction, and Shawshank Redemption, then the two have similar interests. We can logically deduce that X would like Shawshank Redemption and Y would like Interstellar. Collaborative filtering uses user interests and behavior for its recommendations.

For this project, User-User collaborative filtering has been used. The algorithm first finds how similar each user is compared to other users and computes a similarity score. It picks out the most similar users and recommends movies that they have liked previously based on this similarity score.

Taking our movie dataset example, the similarity between users is found by the User-User collaborative filtering algorithm based on ratings they gave to various movies. The prediction of an item for a user is calculated by the weighted sum of the user ratings given by other users to an item.

The prediction Pu,i is given by:

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v s_{u,v}}$$

Where:

- Pu,i is the prediction of an item
- Rv,i is the rating given by a user v to a movie i
- Su,v is the similarity between users

Fig 2.3: Prediction formula

To predict the ratings for other users based on the ratings for users we have, we can do the following steps:

1. Similarity between the user u and v is needed for making predictions. For this, we use Pearson correlation.
2. The next step is to find the correlation value of the two users. We can do this by finding the items rated by both users u and v and find correlation based on that.
3. Next, we calculate the predictions using the similarity scores. This algorithm calculates the similarities between users and then based on each similarity finds out the predictions.
4. Recommendations are made based on these prediction values.

This process consumes a lot of time as it calculates the similarity for each user in a database of many users and then calculates prediction for each similarity score. Selecting only the neighbors to the current user to make predictions is a smart way to handle this problem. Out of the many ways to select neighbors, in this project we have chosen the top-N users with the highest similarity value using the K-nearest neighbors algorithm.

Before implementing these concepts, we need to handle cold starts. A cold start is when a new user or a new item is being added to the database. Cold starts are of two types:

1. Visitor Cold Start – when a new user signs up to the platform and is added to the database. Since there is no past data of that user, the system does not know the interests of that particular user. It becomes hard for the system to recommend movies to them. We solve this problem by recommending the most popular overall movies to the new user until we get more information.
2. Movie Cold Start - when a new movie is launched or introduced to the system. Determining key parameters like ratings, user action is very important, but for new movies, we do not have any user action data. We employ content-based filtering to solve this problem by using the genre of the new movie for recommendations.

3. SYSTEM DESIGN AND OVERVIEW

A. Design Overview:

This project is created using Spring boot, Flask, Hibernate, and JPA with MySQL. JSTL and JPQL were used at some places. 3 Flask APIs (2 for content-based and 1 for collaborative) were written and called onto the JSPs using JavaScript. Data was exchanged in form of JSON during API calls. For handling entities and their attributes, Hibernate was used and corresponding Java Classes were written. Spring core security dependency was used to implement sessions, Authorization, Authentication along with Password Encryption using *BCryptPasswordEncoder*, the latest password encoding technique.

B. Implementation:

A new project was created using Spring *Initializr*. Maven was used as a PMT. Dependencies like Spring web, Spring Data JPA, Lombok, Dev Tools, etc. were added graphically during project creation. Additional dependencies like JSTL, MySQL driver, spring-boot-starter-mail, Gson, tomcat-embed-jasper, spring-boot-starter-security, etc. were added directly in the POM file. Separate packages were created in the project structure to contain service classes, controllers, runner class, model classes, security files, and repositories thus attaining modularity. Appropriate annotations and imports were made inside classes and the necessary credentials were mentioned in the *application.properties* file.

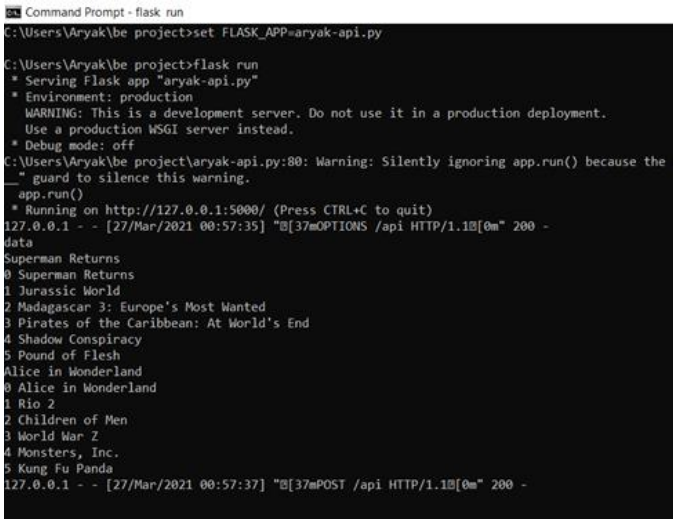
Loading the dataset:
An empty database was created, and tables (with their associated relationships) were generated on the fly by Hibernate. The dataset in the .csv format used read into the spring controller and a request mapping was written to fire the dataset into the DB. On triggering the .csv mapping, the dataset was then loaded into the tables: "movie" and "rating". Inputs from users were appended into the rating table.

For the recommendation part, 3 Flask APIs were created separately: 2 for content-based and one for collaborative filtering. The first content-based filtering API (triggered when the user clicks on the "recommended for me" button) works and recommends movies based on a fixed set of attributes, which cannot be altered by the user. The second API (triggered when a user hits the customize button) expects the user to specify a list of attributes that the user decides as a basis of recommendation. This is a customized recommendation and works the same as the first API if the user does not specify any features for recommendations. Both content-based APIs are POST APIs and expect the user's favorite movie list as input for creating the cosine matrix. After the cosine matrix is created, the most relevant movies are sorted and sent back as a response. The data coming from the API is then displayed on the web page using JSTL.

Multiple request mappings were written in the Spring controller and corresponding JSPs were generated. Login, logout, session management, error pages, and forbidden pages were handled by Spring Security dependency. Authentication, authorization, and URL security were established by writing rules in Security classes.

4. RESULT

Here is the output of the first content-based API. In this case, the user has marked 'Superman Returns' and 'Alice in Wonderland' as his favorite movies. 10 movies similar to these movies have been given as output based on cosine similarity values.

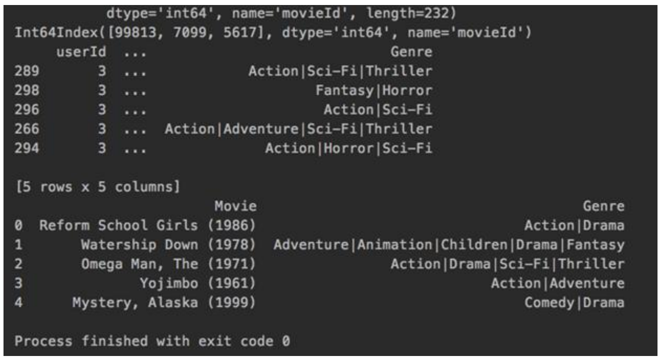


```
C:\Users\Aryak\be project>set FLASK_APP=aryak-api.py

C:\Users\Aryak\be project>flask run
 * Serving Flask app "aryak-api.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
C:\Users\Aryak\be project\aryak-api.py:80: Warning: Silently ignoring app.run() because the
__" guard to silence this warning.
  app.run()
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Mar/2021 00:57:35] "[37mOPTIONS /api HTTP/1.1[0m" 200 -
data
Superman Returns
0 Superman Returns
1 Jurassic World
2 Madagascar 3: Europe's Most Wanted
3 Pirates of the Caribbean: At World's End
4 Shadow Conspiracy
5 Pound of Flesh
Alice in Wonderland
0 Alice in Wonderland
1 Rio 2
2 Children of Men
3 World War Z
4 Monsters, Inc.
5 Kung Fu Panda
127.0.0.1 - - [27/Mar/2021 00:57:37] "[37mPOST /api HTTP/1.1[0m" 200 -
```

Fig 4.1: Content-based filtering output

Below is the output of the collaborative filtering API. The first table is the top 5 movies the user has rated highly. The bottom table shows the 5 highest rated recommendations from the K nearest neighbors to the current user.



```
          dtype='int64', name='movieId', length=232)
Int64Index([99813, 7099, 5617], dtype='int64', name='movieId')
     userId  ...                          Genre
289       3  ...          Action|Sci-Fi|Thriller
298       3  ...                  Fantasy|Horror
296       3  ...                   Action|Sci-Fi
266       3  ...  Action|Adventure|Sci-Fi|Thriller
294       3  ...             Action|Horror|Sci-Fi

[5 rows x 5 columns]
                        Movie                                        Genre
0  Reform School Girls (1986)                                 Action|Drama
1       Watership Down (1978)  Adventure|Animation|Children|Drama|Fantasy
2       Omega Man, The (1971)                Action|Drama|Sci-Fi|Thriller
3              Yojimbo (1961)                             Action|Adventure
4      Mystery, Alaska (1999)                                 Comedy|Drama

Process finished with exit code 0
```

Fig 4.2: Collaborative filtering output

5. CONCLUSION

In this paper, we introduced a web-based system for movie recommendations created using Spring Boot. It expects a user to mark a few favorite movies and then recommends a list of 10 movies based on cosine similarity for content-based filtering. By the nature of our system, it is not possible to judge the performance measure since there is no correct or incorrect recommendation; it is just based on a person's opinions. We tried testing the system for a small audience and received positive feedback from them. While our system also works on the collaborative approach for recommendations, it enables a user to explore what most users find interesting. The ratings dataset plays a crucial role and is the heart of the collaborative approach. The project is a web application created using Spring Boot and Flask APIs which permits a user to give ratings to different movies and also recommends appropriate movies based on other users' ratings and their liking.

REFERENCES

[1] Xie HT, Meng XW. A personalized information service model adapting to user requirement evolution. Acta Electronica Sinica, 2011,39(3):643-648 (in Chinese with English abstract)

[2] Munoz-Organero, Mario, Gustavo A. Ramíez-González, Pedro J. Munoz-Merino, and Carlos Delgado Kloos. "A Collaborative Recommender System Based on Space-Time Similarities", IEEE Pervasive Computing, 2010

[3] https://medium.com/data-science-101/movie-recommendation-system-content-filtering-7ba425ca0920

[4] https://medium.com/code-heroku/building-a-movie-recommendation-engine-in-python-using-scikit-learn-c7489d7cb145

[5] Joseph A. Konstan, John Riedl. Recommender systems: from algorithms to user experience. User Model User-Adapt Interact, March 2012

[6] Chenguang Pan, Wenxin Li. Research Paper Recommendation with Topic Analysis. In Computer Design and Applications IEEE, 2010

[7] Pu P, Chen L, Hu R. A User-Centric Evaluation Framework for Recommender Systems. In: Proceedings of the fifth ACM conference on Recommender Systems, ACM, 2011

[8] Bhavik Pathak , Robert Garfinkel , Ram D. Gopal , Rajkumar Venkatesan & Fang Yin .Empirical Analysis of the Impact of Recommender Systems on Sales, Journal of Management Information Systems, December 2014

[9] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, John Riedl . Getting to Know You: Learning New User Preferences in Recommender Systems. In: Proceedings of the international conference on intelligent user interfaces, 2002

[10] J. Ben Schafer, Joseph Konstan, John Riedl. Recommender Systems in E-Commerce. In: Proceedings of the 1st ACM conference on electronic commerce, 1999.

[11] A. Merve Acilar, Ahmet Arslan. A collaborative filtering method based on artificial immune network. Expert Systems with Application, 2009

[12] Chen LS, Hsu FH, Chen MC, Hsu YC. Developing recommender systems with the consideration of product profitability for sellers. Int J Inform Sci 2008.

[13] Jalali M, Mustapha N, Sulaiman M, Mamay A. WebPUM: A Web-based recommendation system to predict user future movements.Exp Syst Applicat, March 2010

[14] Adomavicius G, Tuzhilin A . Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE, 2005.