JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

MOVIE REVIEW SYSTEM

[1]Kirti Pal, [2]Insha Siddiqui, [3]Dr.Pankaj Kumar

[1]Student, [2]Student, [3]Assistant Professor

[1]Computer Science and Engineering, Shri Ramswaroop Memorial College of Engineering and Management, Lucknow, India,
[2]Computer Science and Engineering, Shri Ramswaroop Memorial College of Engineering and Management, Lucknow, India,
[3]Computer Science and Engineering, Shri Ramswaroop Memorial College of Engineering and Management, Lucknow, 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.

IndexTerms **- Recommender System, Machine Learning, Collaborative Filtering, Content based Filtering**

1. INTRODUCTION

Recommender systems are have emerge as a staple in this era of the net economy. They assist in decreasing the overload of statistics through offering custom designed statistics access. Modeling, programming, and deploying those recommender structures have allowed agencies to decorate sales and maintain customers. These structures consists of custom designed seek engines, personalized buying agents, and hand- made content material indices. The scope of personalization and use of those structures extends to many specific areas, now no longer simply internet pages. The set of rules and ideas utilized in recommender systems can range from keyword matching in consumer profiles, content-primarily based totally filtering, collaborative filtering, to extra state-of-the-art strategies of statistics mining inclusive of clustering server logs. Recommendation structures clear out facts that the use of diverse standards and procedures and endorse the maximum applicable gadgets to customers primarily based totally on customizable criteria. It first captures the past behaviour of a patron and recommends merchandise primarily based totally on that. This proves to be beneficial to the person and customers as customers simplest see applicable content material and are not bombarded with useless merchandise and customers get higher engagement. Hence it come to be vital to construct clever advice structures and employ the beyond conduct in their customers.

2. DOMAIN OVERVIEW

2.1.Content-based filtering

The content-based filtering approach attempts to take a look the liking of a person given the movies' features, which the person reacts positively to it. Content refers back to the capabilities or attributes of the movies the person likes. The gist of this method of advice is to examine movies the use of unique attributes, recognize what the person has already liked, predicting what he may also like, and recommend a some movies from the database with the maximum comparable attributes. These attributes can be constant or unique with the aid of using the person himself. This approach of recommendation uses movie features to suggest other similar movies to what the customers has already liked.

This method entails 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 preceding records. Then, the pinnacle few movies which are most similar are advocated to the user.

User's feedback and its importance in recommendation:

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

Explicit Feedback: The users specify their liking via means of actions like reacting to an item, marking it as their favorite, or rate it. In our system, the user marks positive movies as 'favorites' and additionally offers ratings to movies. The ratings provided are used for collaborative filtering.

Cosine Similarity:

It is a concept used to measure how comparable one movie is to each different movie within the dataset. It measures the cosine of the perspective among vectors projected in a multidimensional space. Even if the two similar movies are a ways aside via the Euclidean distance, they will nevertheless be oriented nearer together.

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

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.

2.2.Collaborative-based Filtering

Collaborative filtering is commonly used for records that cannot without difficulty be defined with the aid of using attributes from the dataset inclusive of books and movies. First, the prediction approach builds a user-item matrix of possibilities for items by users. It then fits users by the aid of using computing the similarities among their facts and makes recommendations..

Fig. 2.2: Collaborative Filtering

The following instance allows us apprehend this idea better. If character X likes the three movies - Interstellar, Shutter Island, Extinction, and person Y likes Shutter Island, Extinction, and Shawshank Redemption, then the two have similar pastimes. We can logically deduce that X would love Shawshank Redemption and Y would love Interstellar. Collaborative filtering makes use of people pastimes and conduct for its recommendations.

For this undertaking, User-User collaborative filtering has been used. The set of rules first well-known shows how similar everyone is in comparison to exclusive clients and computes a similarity score It picks out the most similar users and recommends movies that they have preferred formerly previously based totally on this similarity score.

Taking our movie dataset example, the similarity among users is observed by the User-User collaborative filtering algorithm based on ratings they gave to diverse movies. The prediction of an object for a person is calculated with the aid of using the weighted sum of the person rankings given with the aid of using different customers to an object.

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 among the user u and v is needed for making predictions. For this, we use Pearson correlation.

2. The subsequent step is to discover the correlation cost of the two users. We can do that by locating the objects rated by each users u and v and discover correlation primarily based totally on that.

3. Next, we calculate the predictions with the use of similarity scores. This set of rules calculates the similarities among users after which primarily based totally on every similarity reveals out the predictions.

This method consumes a lot of time as it calculates the similarity for every person in a database of many users after which calculates prediction for each similarity score. Selecting handiest the friends to the contemporary person to make predictions is a clever manner to deal with this problem. Out of diverse techniques to choose buddies, in this task we have got were given decided on the top-N clients with the quality similarity fee using the K-nearest buddies set of rules.

Before imposing these concepts, we need to handle cold starts. A cold start is when a new consumer or a new object is being brought to the database. Cold starts are of two types:

1. **Visitor Cold Start** –Since there is no beyond information of the purchaser the system does no longer understands the intrests precise purchaser. It becomes tough for the machine to propose movies to them. We resolve this trouble with the aid of using recommending the maximum famous common films to the brand new consumer till we get extra information.

2. **Movie Cold Start** - when a new movie is released or delivered to the system. Determining key parameters like ratings, consumer motion could be very important, however for brand new movies, we do now no longer have any consumer motion data. We hire primarily based totally filtering to clear up this trouble via way of means of the usage of the style of the brand new movie film for recommendations..

3. SYSTEM DESIGN AND OVERVIEW

3.1.Design Overview:

This project is created with using of Spring boot, Flask, Hibernate, and JPA with MySQL. JSTL and JPQL had been used at few places. 3 Flask APIs (2 for content-based and 1 for collaborative) had been written and referred to as onto the JSPs with the use of JavaScript. Data were exchanged in form of JSON all through API calls. For managing entities and their attributes, Hibernate came in use and corresponding Java Classes had been written. Spring core security dependency was used to implement sessions, Authorization, Authentication along with Password Encryption using the latest password encoding idea.

3.2. Implementation:

A new project was created with the use of Spring Initializer. Maven had been used as a PMT. Interdependencies like Spring web, Spring Data JPA, Lombok, Dev Tools, etc. were delivered graphically at some point of project creation. Additional dependencies like JSTL, MySQL driver, spring-boot-starter-mail, Gson, tomcat embed-jasper, spring-boot-starter-safety, etc were delivered right now withinside POM file. Separate packages were created withinside the assignment shape to contain carrier trainings, controllers, runner class, model trainings, protection files, and repositories as a end result reaching standard. Appropriate annotations and imports have been made interior instructions and the essential credentials have been mentioned.

Loading the dataset:

An empty database changed into created, and tables (with their related relationships) had been generated at the fly with the aid of using Hibernate. The data collected in the .csv layout used study into the spring controller and a request mapping changed into written to fire the dataset into the DB. On activating the .csv mapping, the dataset changed into loaded into the tables: "movie" and "rating". Inputs from customers had been appended into the rating table.

For the recommendation part, three Flask APIs were created separately, two for content-based and one for collaborative filtering. The first content-based filtering API (activated 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 (activated 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 maximum applicable movies are taken care of and dispatched again as a response. The records coming from the API is then displayed at the net web page with the use of JSTL.

Multiple request mappings had been written withinside the Spring controller and corresponding JSPs were generated. Login, logout, consultation management, error pages, and forbidden pages had been dealt with the aid of using Spring Security dependency. Authentication, authorization, and URL safety had been installed with the aid of using writing guidelines in Security classes.

4. RESULT

.

It will recommend top 10 similar movies from the database.

```
def get_movie_recommendation(movie_name):
    n_movies_to_reccomend = 10
    movie_list = movies[movies['title'].str.contains(movie_name)]
    if len(movie_list):
        movie_idx= movie_list.iloc[0]['movieId']
        movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
        distances , indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccome
        rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolis
        recommend_frame = []
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index
            recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':val[1
        df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
        return df
    else:
        return "No movies found. Please check your input"
```

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.

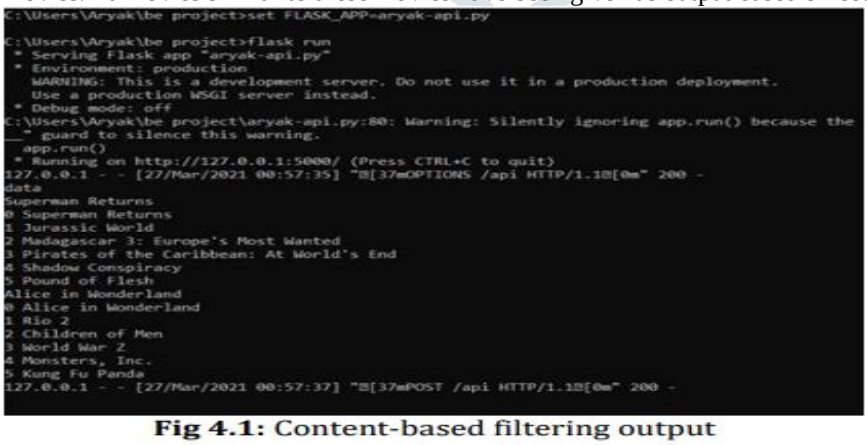

Fig 4.1: Content-based filtering output

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.

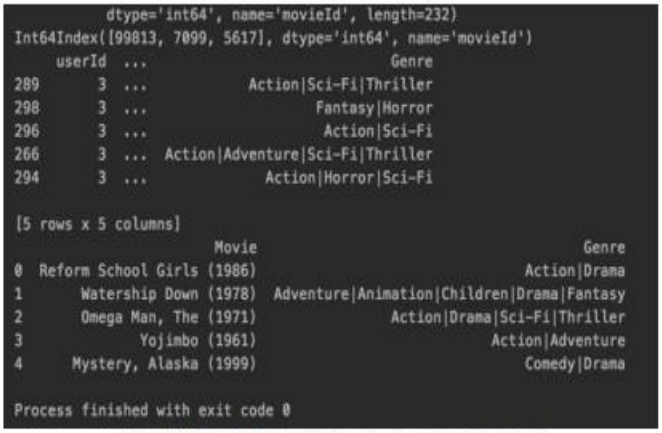

Fig 4.2: Collaborative filtering output

5. CONCLUSION

In this paper, we added a web-primarily based totally gadget for movie recommendations created with the use of 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 performs an important role and is the coronary heart of the collaborative approach. The undertaking is an internet utility created the use of Spring Boot and Flask APIs which allows consumer to provide rankings to exclusive movies and additionally recommends suitable movies primarily based totally on different users rankings and their likings.

REFRENCES

[1] Olivier C., Blaise Pascal University: "Neural network modeling for stock movement prediction, state of art".

2007 .

[2] Leng, X. and Miller, H.-G.: "Input dimension reduction for load forecasting based on support vector machines", IEEE International Conference on Electric Utility Deregulation, Restructuring and Power Technologies (DRPT2004), 2004.

[3] XiZhang1, Siyu Qu1, Jieyun Huang1, Binxing Fang1, Philip Yu2, "Stock Market Prediction via Multi-Source Multiple Instance Learning." IEEE 2018.

[4] Kang Zhang et al. "Stock Market Prediction Based on Generative Adversarial Network", Procardia Computer Science, 147(2019).

[5] M. Usmani, S. H. Adil, K. Raza and S. S. A. Ali, "Stock market prediction using machine learning techniques," 2016 3rd International Conference on Computer and Information Sciences(ICCOINS), Kuala Lumpur, 2016