MOVIE RECOMMENDATION SEARCH ENGINE

Presented By

KISHORE RAJ ETHIRAJ

RACHIT CHOKSI

Contents

1. INTRODUCTION

Recommendation system has become very popular in many aspects in real social networks, such as e-commerce services Amazon.com, movie rating website IMDB, and DVD rental service company Netflix. This project focuses on multiple level link prediction of recommending movies. Movie rating data has become largely accessible via the Internet. Most rating systems are designed to have 5 score choices from 1 to 5. However, in finding a potential interesting movie for a certain user, there is not much difference between score 4 and 5. In fact, this is a classification problem on predicting the category of a movie that has not been watched by the user. So instead of a specific score, we are more concerned with the genre of the movie and based on the user input the code will present the end user with the desirable results. Another important motivation for this project is to build a social recommendation system in the future. The idea comes from recommendations we get from our friends in everyday life. It is quite common that we tend to value more about reviews given from other strangers before watching. In this project, we have used a data set consisting of thousands of movie titles which are saved along with multiple genres along with the ratings. So once the user selects three fields which are genres, the recommendation system will give the user the best results matching all the genre which are having the best rating in a sorting order.

2. EXISTING SYSTEM

Problem Identification:

The biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations. It's no coincidence that the companies most identified with having excellent recommendations are those with a lot of consumer user data: Google, Amazon, Netflix, Last.fm. The more item and user data a recommender system must work with, the stronger the chances of getting good recommendations. But to get good recommendations, you need a lot of users, so you can get a lot of data for the recommendations. Another problem with search engine would be the rapidly changing data. Clearly an algorithmic approach will find it difficult if not impossible to keep up with fashion trends. Lastly the search engine is unpredictable because it cannot give a perfect match every time.

3. PROPOSED SYSTEM

The Movie Recommendation Search Engine uses a user based collaborative filter. User based CF approach is very like item based CF. But in this one, similarity is computed between users, not item. The underlying assumption of the collaborative filtering approach is that if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. The search engine will take the user's input on genres and movie name and recommends the user a list of 10 movies that he would also enjoy. The search engine uses k means clustering method to split the object in the dataset into clusters. A class label will be attached to the clusters for matching based on the user input. A hard code on the search engine is that the user's input for movie name should be in the order of 5,4,3 ratings since the search engine would return a movie based on the hard code.

4. LITERATURE REVIEW

The term "collaborative filtering" was introduced in the context of the first commercial recommender system, called Tapestry, which was designed to recommend documents drawn from newsgroups to a collection of users. The motivation was to leverage social collaboration to prevent users from getting inundated by a large volume of streaming documents. Collaborative filtering, which analyzes usage data across users to find well matched user-item pairs, has since been juxtaposed against the older methodology of content filtering which had its original roots in information retrieval. In content filtering, recommendations are not "collaborative" in the sense that suggestions made to a user do not explicitly utilize information across the entire user-base. Some early successes of collaborative filtering on related domains included the Group Lens system.

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

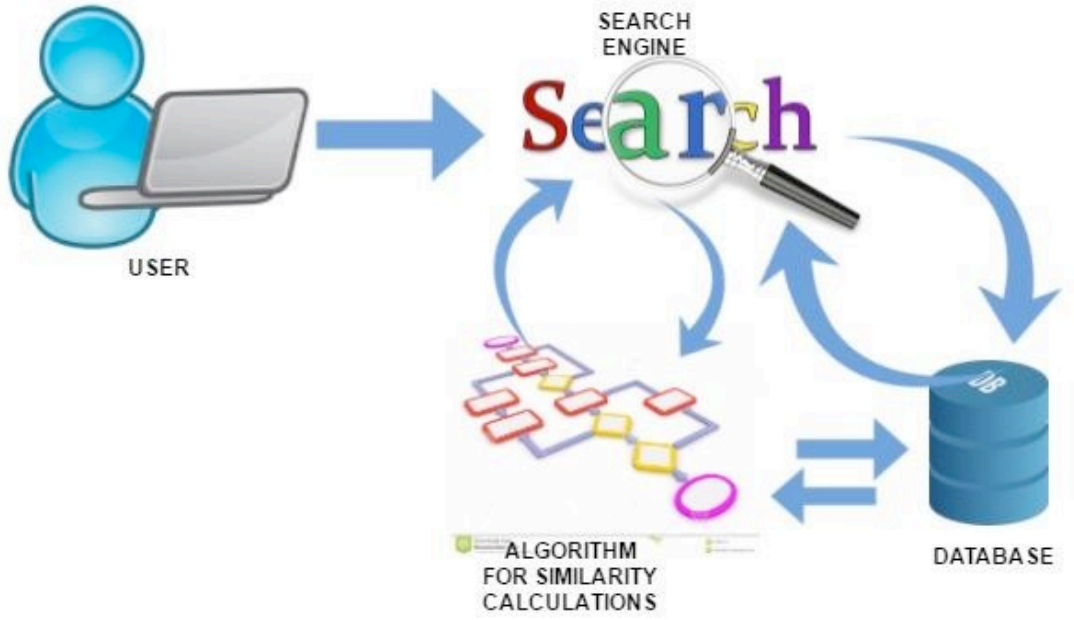
1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

Over the last decade, lots of researchers have studied innovative approaches of recommender systems to solve these problems of CF (Collaborative Filter method) and CB (Content Based Filtering method), and to apply them into real world problems. Especially, applications of

data mining techniques to recommender systems have been effective to offer personalized information to the user through analysis of his/her preference. In the past, many have used Item Based Collaborative Filtering for their respective search engine. But with an increasing data each day, the search engine should be accurate and the database should be updated often. We use a User Based Collaborative Filter which computes k means algorithm to find the nearest neighbor within the cluster. By using k means clustering algorithm, the accuracy of the search engine improves.

5. METHODOLOGY

5.1 Architecture:



5.2 Algorithm

This approach uses user rating data to compute the similarity between users or items. This is used for making recommendations. This was an early approach used in many commercial systems. It's effective and easy to implement. Typical examples of this approach are neighborhood-based CF and item-based/user-based top-N recommendations. For example, in user based approaches, the value of ratings user 'u' gives to item 'I' is calculated as an aggregation of some similar users' rating of the item:

$$r_{u,i} = \text{aggr}_{u' \in U} r_{u',i}$$

where 'U' denotes the set of top 'N' users that are most similar to user 'u' who rated item 'i'. Some examples of the aggregation function includes:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

$$r_{u,i} = k \sum_{u' \in U} \text{simil}(u,u') r_{u',i}$$

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \text{simil}(u,u')(r_{u',i} - \bar{r_{u'}})$$

where k is a normalizing factor defined as $k = 1/\sum_{u' \in U} |\text{simil}(u,u')|$. and $\bar{r_u}$ is the average rating of user u for all the items rated by u.

The neighborhood-based algorithm calculates the similarity between two users or items produces a prediction for the user by taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach. Multiple measures, such as Pearson correlation and vector cosine based similarity are used for this.

The Pearson correlation similarity of two users x, y is defined as

$$\text{simil}(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

where I_{xy} is the set of items rated by both user x and user y.

The cosine-based approach defines the cosine-similarity between two users x and y as:[4]

$$\text{simil}(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

The user based top-N recommendation algorithm uses a similarity-based vector model to identify the k most similar users to an active user. After the k, most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. A popular method to find the similar users is the Locality-sensitive hashing, which implements the nearest neighbor mechanism in linear time.

The advantages with this approach include: the explain ability of the results, which is an important aspect of recommendation systems; easy creation and use; easy facilitation of new data; content-independence of the items being recommended; good scaling with co-rated items.

There are also several disadvantages with this approach. Its performance decreases when data gets sparse, which occurs frequently with web-related items. This hinders the scalability of this approach and creates problems with large datasets. Although it can efficiently handle new users because it relies on a data structure, adding new items becomes more complicated since that

representation usually relies on a specific vector space. Adding additional items requires inclusion of the additional item and the re-insertion of all the elements in the structure.

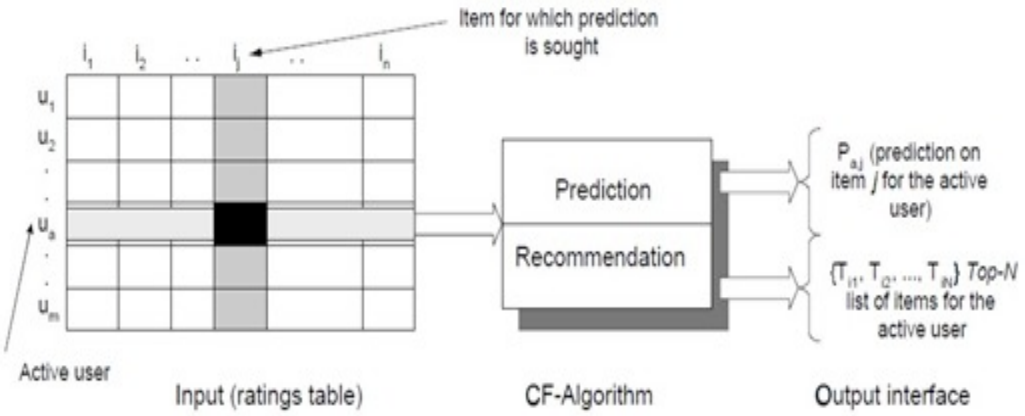


Figure 1 The collaborative filtering process.

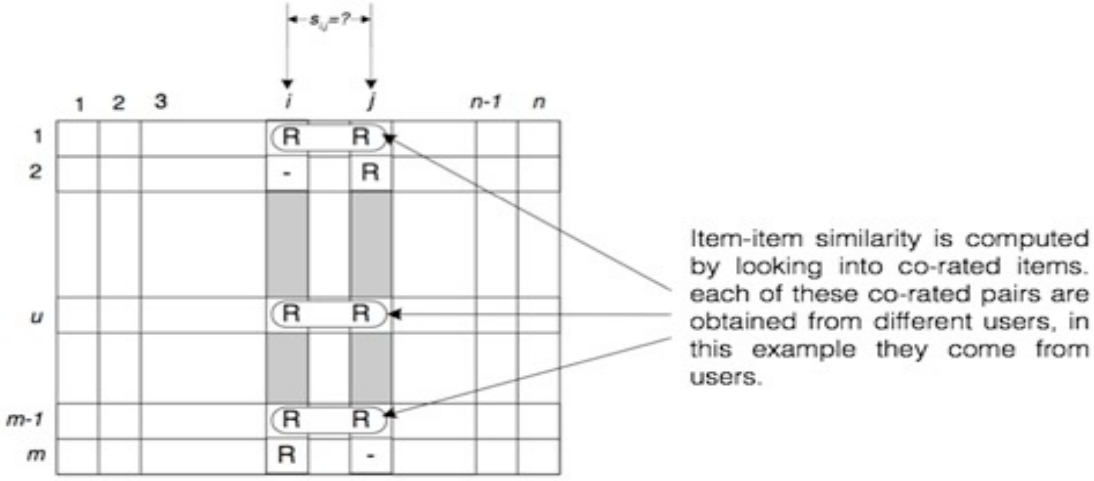


Figure 2: Isolation of the co-rated items and similarity computation.

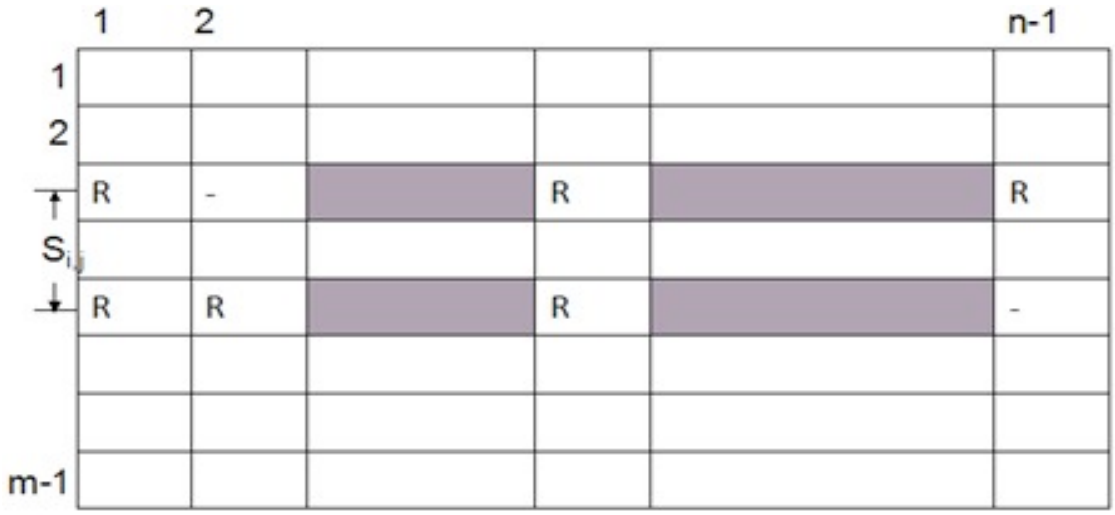


Figure 3: Isolation of the co-rated users and similarity computation.

K means algorithm:

Algorithm: k-means. The k-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

k: the number of clusters,

D: a data set containing n objects.

Output: A set of k clusters.

Method:

(1) arbitrarily choose k objects from D as the initial cluster centers;

(2) repeat

(3) (re)assign each object to the cluster to which the object is the most similar,

based on the mean value of the objects in the cluster;

(4) update the cluster means, that is, calculate the mean value of the objects for

each cluster;

(5) until no change;

Libraries used:

- library(proxy): Distance and Similarity Measures
  Provides an extensible framework for the efficient calculation of auto- and cross-proximities, along with implementations of the most popular ones.
- library(recommenderlab): Lab for Developing and Testing Recommender Algorithms
  Provides a research infrastructure to test and develop recommender algorithms including UBCF, IBCF, FunkSVD and association rule-based algorithms.
- library(reshape2): A Reboot of the Reshape Package
  Flexibly restructure and aggregate data using just two functions: melt and 'dcast' (or 'acast').
- library(ggplot2): Create Elegant Data Visualizations Using the Grammar of Graphics
  A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.
- library(shiny): Web Application Framework for R
  Makes it incredibly easy to build interactive web applications with R. Automatic "reactive" binding between inputs and outputs and extensive pre-built widgets make it possible to build beautiful, responsive, and powerful applications with minimal effort.
- library(shinythemes):
  Themes for use with Shiny. Includes several Bootstrap themes from, which are packaged for use with Shiny applications

6. IMPLEMENTATION

6.1 System Design

System design is transition from a user oriented document to programmers or database personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

6.2 Sample Codes

- Dataset for Movie Recommendation Search Engine is in .csv format. So, we have used below code to read data in R studio.

```
search <- read.csv("search.csv", stringsAsFactors=FALSE)
ratings <- read.csv("ratings.csv", header = TRUE)
search <- search[-which((search$movieId %in% ratings$movieId) == FALSE),]
```

- Sample dataset screenshot of Movie dataset

	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance
8	8	Tom and Huck (1995)	Adventure|Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action|Adventure|Thriller

- To use the ratings data for building a recommendation engine with recommenderlab, I convert rating matrix into a sparse matrix of type realRatingMatrix.

```
ratingmatrix <- dcast(ratings, userId~movieId, value.var = "rating", na.rm=FALSE)
ratingmatrix <- ratingmatrix[,-1]
colnames(userSelect) <- colnames(ratingmatrix)
ratingmatrix_new <- rbind(userSelect,ratingmatrix)
ratingmatrix_new <- as.matrix(ratingmatrix_new)

#Convert rating matrix into a sparse matrix
ratingmatrix_new <- as(ratingmatrix_new, "realRatingMatrix")
```

- Output:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	4.0	NA	NA	NA	NA	NA	NA	NA	1.5	NA	NA	NA	NA	NA	NA	NA	4.0	NA	NA	NA
3	5.0	NA	2.0	NA	3.0	NA	NA	NA	NA	NA	NA	NA	NA	4	NA	NA	5.0	NA	NA	NA	NA	NA	NA	NA	3.0	NA	NA	NA	NA	NA	NA	3.0	NA	NA	NA
4	NA	NA	NA	NA	3.0	NA	3.0	NA	NA	NA	4.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA
5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	4	NA	5.0	NA	NA	NA
6	4.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	NA	NA	NA	NA	NA	NA	NA	NA	NA	4.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA	NA	NA
9	5.0	NA	4.0	NA	3.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA	NA	NA	NA	NA	NA	3.0	NA	NA	NA
10	NA	NA	3.0	NA	NA	4.0	NA	NA	NA	3.0	NA	NA	NA	3	NA	4.0	NA	NA	2.0	NA	3.0	NA	3.0	NA	4.0	NA	NA	NA	NA	NA	NA	3.0	NA	NA	NA
11	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
12	4.0	NA	NA	NA	NA	NA	NA	NA	NA	3.0	NA	NA	NA	NA	NA	NA	3.0	NA	3.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	3.0	NA
13	NA	NA	3.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	1.5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
14	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	3.5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
15	4.0	NA	3.0	NA	3.0	NA	3.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
16	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
17	NA	3.0	NA	NA	NA	NA	NA	NA	4	NA	NA	NA	NA	NA	NA	NA	NA	NA	3.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA	NA	5.0	NA	NA	NA
18	5.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	5.0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

- The recommenderlab package contains some options for the recommendation algorithm:

```
#Create Recommender Model. "UBCF" stands for user-based collaborative filtering
recommender_model <- Recommender(ratingmatrix_new, method = "UBCF",param=list(method="Cosine",nn=5))
recommender <- predict(recommender_model, ratingmatrix_new[1], n=10)
recom_list <- as(recommender, "list")
no_result <- data.frame(matrix(NA,1))
recom_result <- data.frame(matrix(NA,10))
```

- **We have defined static array to display genres in dropdown.**

- **Sample shiny code to display output**

```
output$table <- renderTable({
  movie_recommendation(input$select, input$select2, input$select3)
})

output$dynamic_value <- renderPrint({
  c(input$select,input$select2,input$select3)
})
```

4.3 UML Diagrams

User Model View:

This view represents the system from the users' perspective.

The analysis representation describes a usage scenario from the end-user's perspective.

Structural View

In this model, the data functionality is arrived from the inside the system.

This model view models the static structures.

Behavioral View

It represents the dynamic of behavioral as parts of the system depicting the interactions of collection between various structural elements described in the user module and structural model view.

Implementation Model View

In this structural and behavioral as parts of the system are represented as they are to be built.

Environmental Model View

In this structural and behavioral aspects of the environment in which the system is to be implemented are represented.

Relationships in UML

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

6.2.1 Class diagram:

Class diagram, one of the most commonly used diagrams in object-oriented system, models the static design view for a system. The static view mainly supports the functional requirements of a system – the services the system should provide to the end users. We will see from our practical experience that lots of fun comes out when modeling out system with class diagrams. The discussion on different views of class diagrams for the system will be put into emphasis later in this paper. A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams involve global system description, such as the system architecture, and detail aspects such as the attributes and operations within a class as well. The most common contents of a class diagram are:

- Classes
- Interfaces
- Collaborations
- Dependency, generalization, and association relationships
- Notes and constraints

Below is a high-level class diagram for the Movie Recommendation System. This diagram depicts the relationship between user, ratings, Movie, Genre. The multiplicity is also shown to help understand the system better. One user can rate several movies and must select at least one and a maximum of three genre and movies to make the search engine work, a movie can belong to several genres at a time, a movie can be rated by number of users,

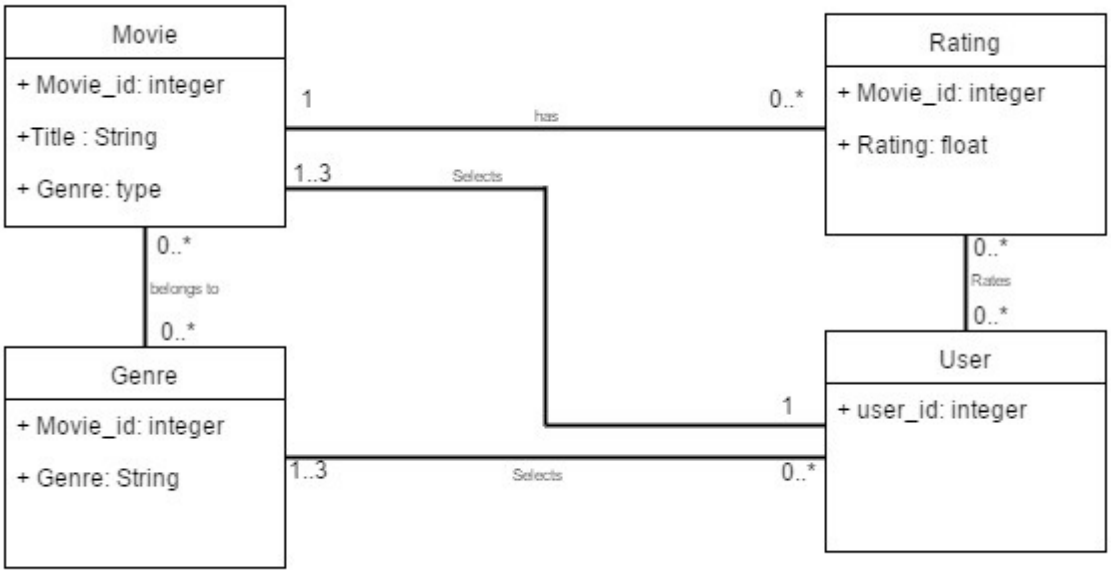
In the following subsections, four groups of class diagrams are presented and analyzed in detail.

Movie: This class contains all the attributes required for identifying a movie uniquely it contains attributes like movie_id, title, genre.

Rating: This class contains the ratings given by different users for different movies.

User: This class contains the user_id attribute and user is responsible for selecting the genre and movies for getting the required movies.

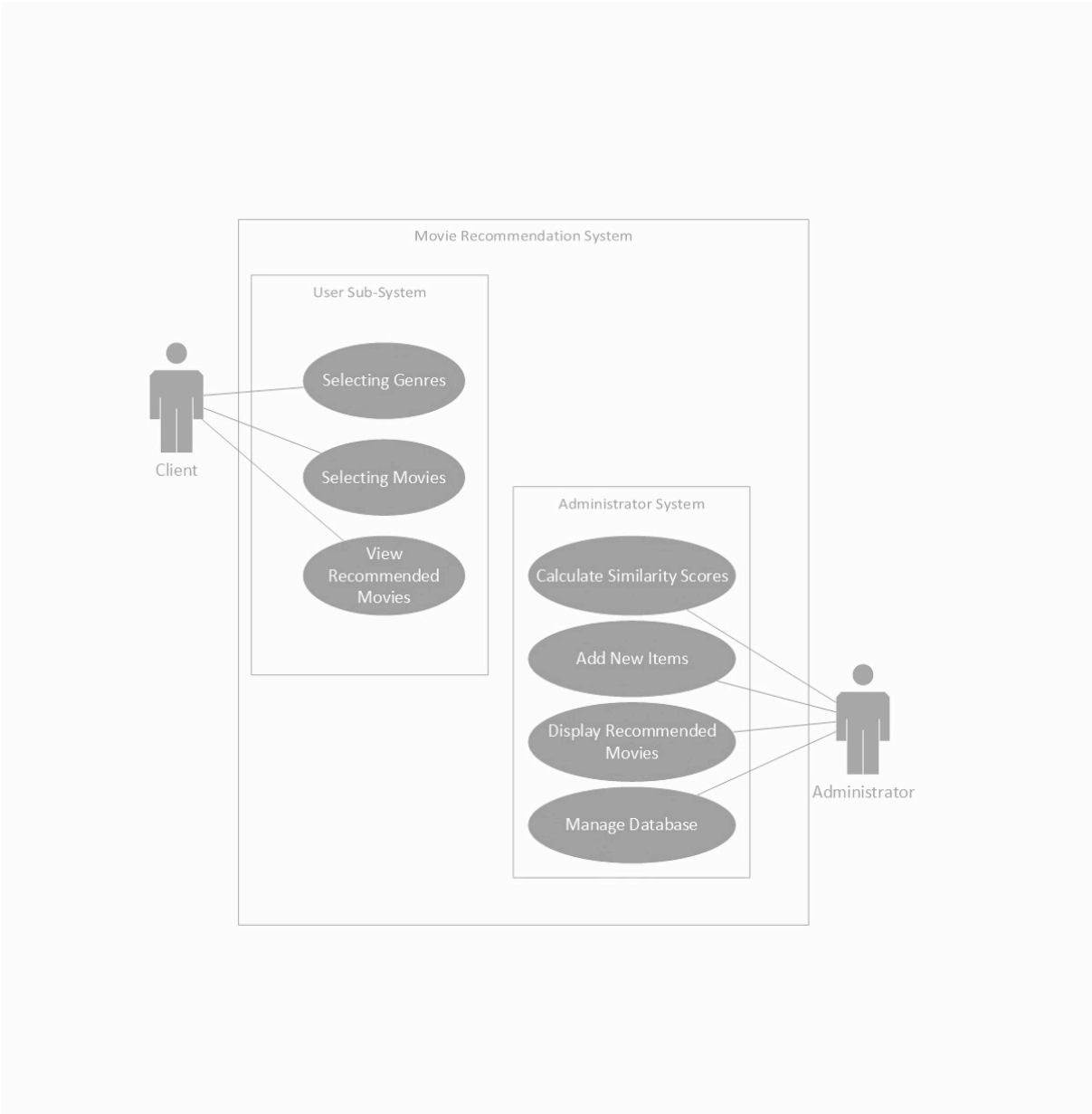
Genre: This class contains the genres to which a movie belongs to.



6.2.2 Use Case Diagram:

Use case diagram contains two actors as shown in the below figure. User is an actor who performs the actions such as selecting genres and based on the Genres selected the recommender system will display the movie list drop down. User has an option to select another movie from the dropdown list to get a movie recommendation of his personal favorite. Once user selects all the mandatory genre fields and confirms the movies displayed for each genre the end user will be displayed with the desired movies list based on the Similarity scores and movies will be displayed in a sorted order.

Admin is another actor who performs actions like updating the movie database, refactoring the algorithm to use indexing or Hash functionalities to decrease the time to retrieve the search results from database.



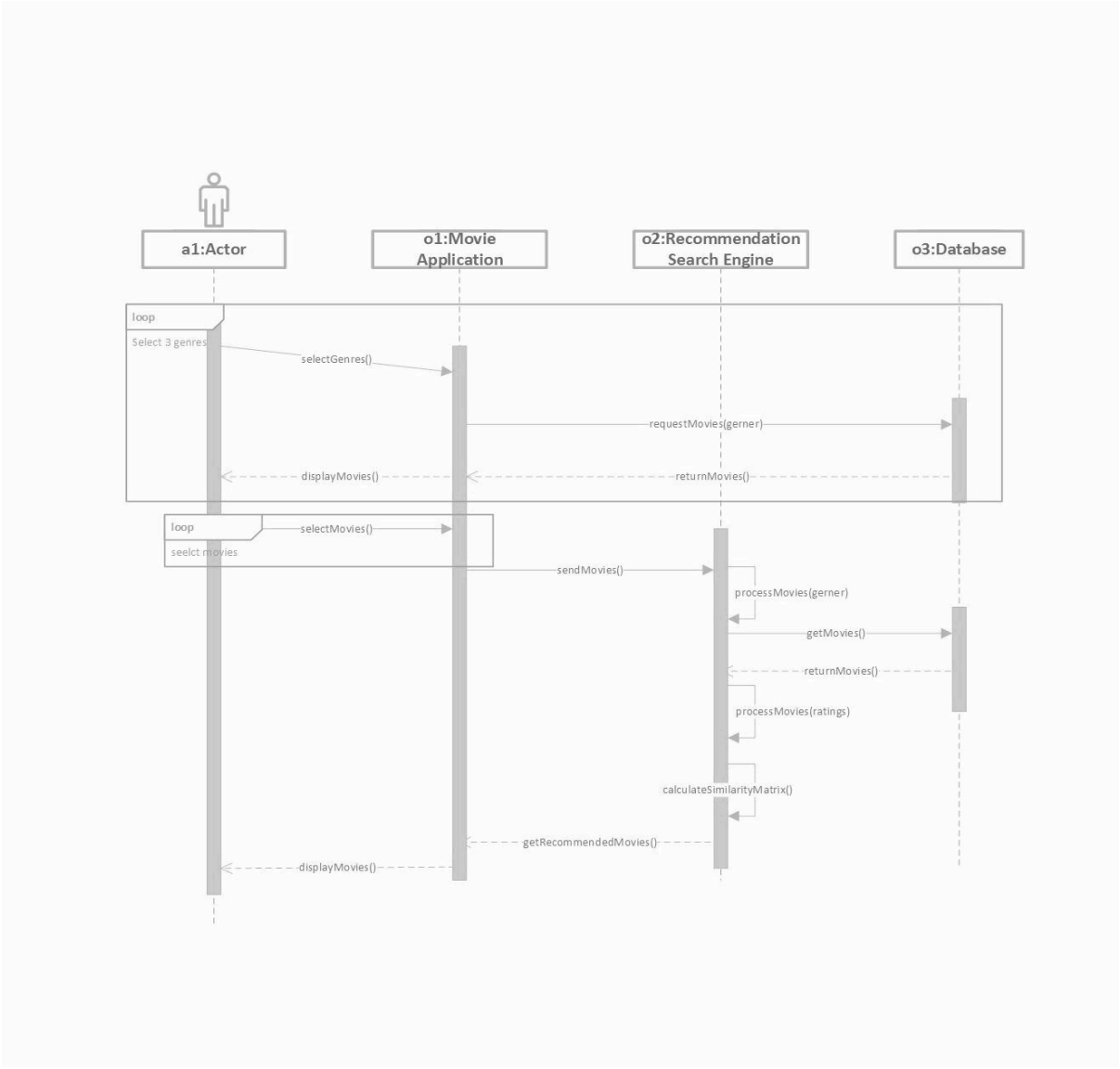
6.2.3 Sequence Diagram:

A sequence diagram is a form of interaction diagram which shows objects as lifelines running down the page, with their interactions over time represented as messages drawn as arrows from the source lifeline to the target lifeline. Sequence diagrams are good at showing which objects communicate with which other objects.

The user object Selects the movie genre from the dropdown list. The movie application receives the select genre request and retrieves the titles from the database which are returned to the movie application. Movie application will display these results to the user object. The above sequence is performed three times in terms of a loop as there are three genre fields user has a select and each loop will perform the same functionality of requesting the movie, database

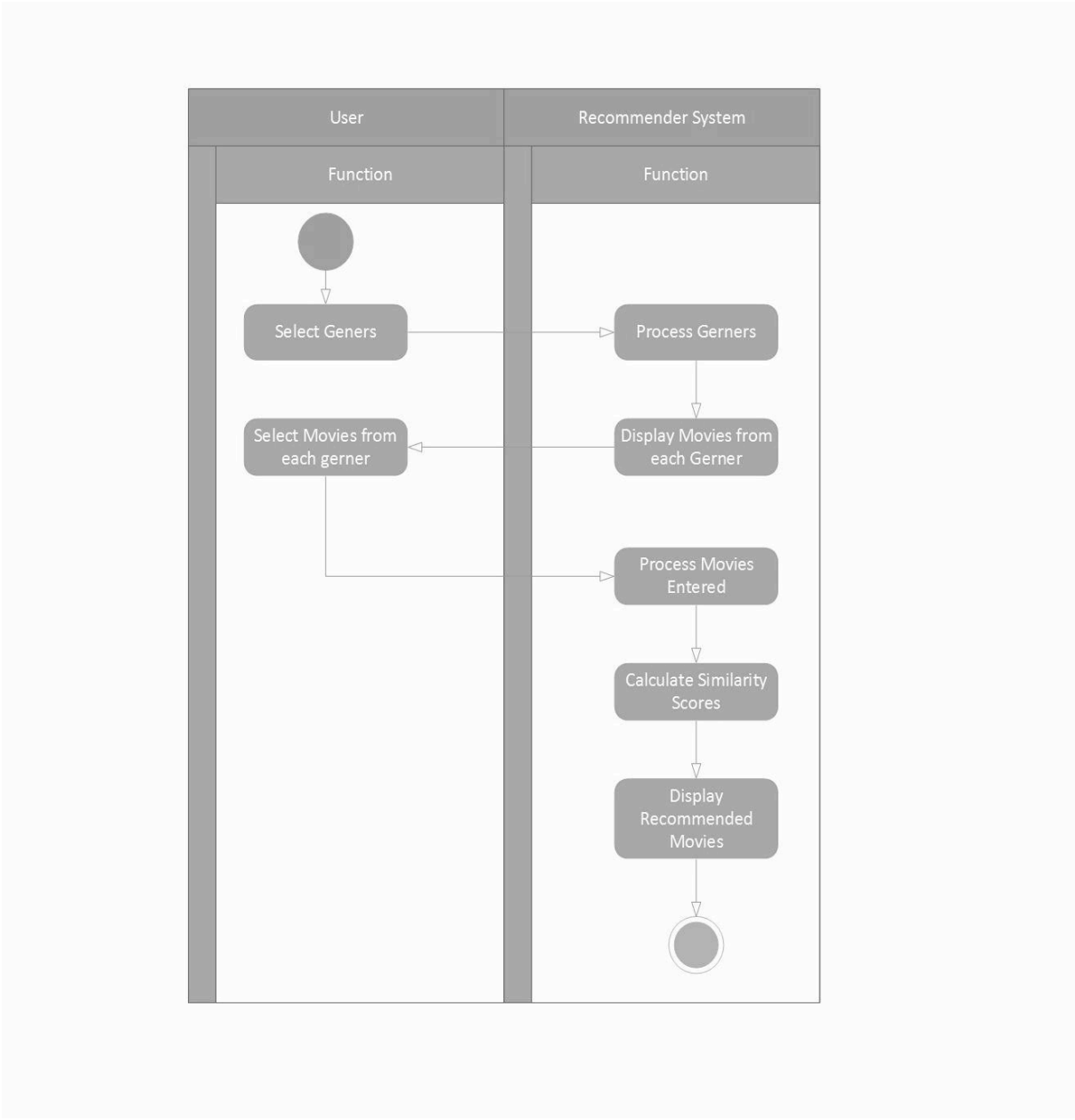
retrieves the movie titles based on request and return them to movie application which will in turn be displayed to the user.

Once the movie is displayed based on user selection, user will have an option to update the movie that is displayed or select another movie from the movie dropdown list. If user changes the movie then the K means algorithm will run and provide movies list based on the similarity. As there are three genres fields the above functionality will be performed multiple times and will provide the End user with the recommended movies list.



6.2.4 Activity Diagram:

Below is the Activity Diagram for Movie Recommendation Search Engine. In this diagram, we have a User and Search Engine. End user is displayed with Select genre field and based on the selected genre the Recommender system will suggest a movie with the top rating available on the Database. End user has will have two options, User can manually select another movie title from the dropdown displayed for the earlier genre selected or use the movie which was selected by Recommender Search Engine. Admin uses the following use cases, receive processing genre requests, displaying top rated movies for each genre selected, If the movie title is changed Recalculate the similarity score and updated the recommended movie list.



6.2.5 Component Diagram:

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

So, from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

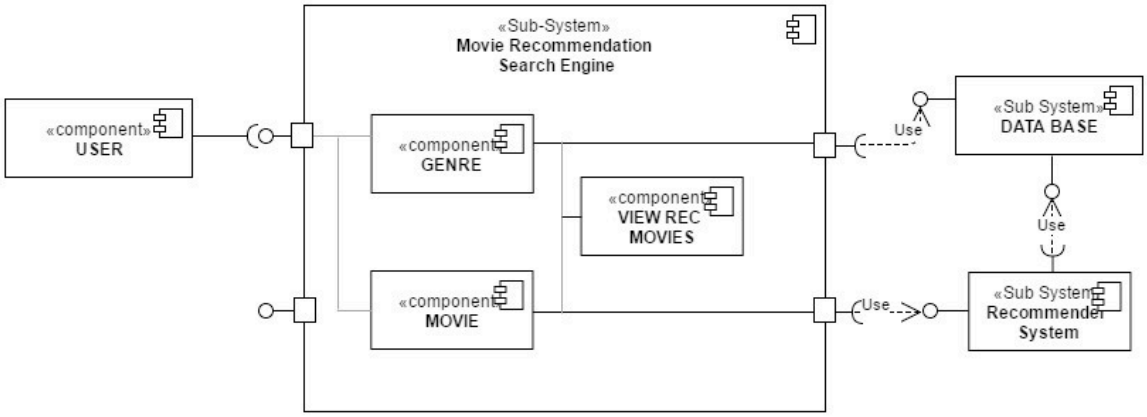
So, the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

The below diagram depicts the component diagram of our Movie Recommender System.

The major components that are present in this system are User, Web Application, Recommender System and the Movie Database.

The web application component provides the interface for the user component to interact with the system. The web application depends on the Recommender System which contains various algorithms for processing the data and both the recommender system and the web application requires the database for their working, so these are the required components in the recommender system implemented.



6.2.6 Deployment Diagram:

The name *Deployment* itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

So, most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

The below diagram depicts the deployment diagram for the movie recommendation system we implemented.

The major nodes in this deployment diagram are the

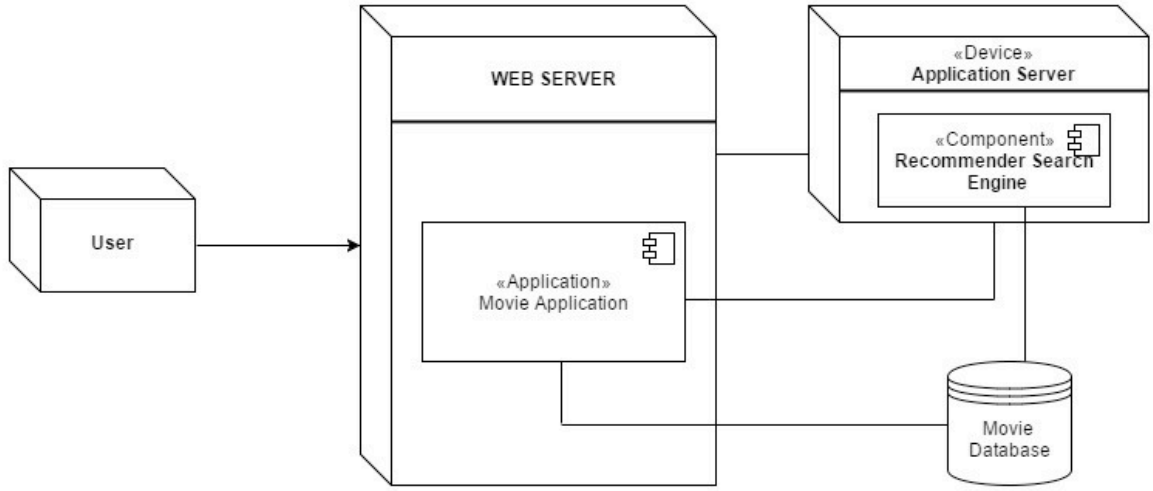
- Client
- Web application,
- Recommender system and
- Movie Database.

The client is nothing but the user who uses the recommendation search engine.

The Web application is the major part by which the user interacts with the search engine we have user Shiny for this application building.

The Recommender System contains different algorithms for calculating the similarity scores and this requires the data which is provided by the database we have used R programming for calculating the similarity scores.

The database contains all the data for making the recommender system work properly and it mainly contains movies, genre and their ratings.



6.3 Project Modules:

System requirement give an idea about what are the necessary things that are needed for the system, which plays a very important role in the development of any system. This chapter deals with what are the hardware components that are needed for the system, application software that is required for the development of the system and the functional requirement of the system.

Frontend tools help to visualize the system, while backend helps in activities which are not visible to the end user.

After the system has been designed physically in detail, the stage is to transfer the system into a working one. Implementation is the stage of a project during which the design of a system is tested, debugged and made operational. So, it is the most crucial stage in achieving a successful new system and in giving the users confidence that the new system will work and be effective.

User:

A common approach to improve reliability and other QoS parameters of a service composition is by dynamic service selection at run time. In a dynamic service composition, a set of functionally equivalent services exists for each service invocation and actual services are incorporated into the execution configuration depending on their most recent QoS parameters. However, two dominant issues limit the application of dynamic compositions on a larger scale: service selection and detection of equivalent services. Since service selection at run time is bonded by additional constraints, like statefulness and composability, state based reliability models need to be applied. However, such models are prone to state explosions, making it difficult to support more complex compositions. The other commonly used approach treats service selection as an optimization problem.

Admin:

This module is to propose an iterative reliability improvement method for service compositions based on the extension of our previous work in. The method consists of: reliability estimation, weak point recommendation and weak point strengthening steps, as defined by the overview. In the rest of this section, we briefly describe each of the stated steps.

Accept user:

The admin can accept the new user request and return the output to the users.

6.4 Software Requirements:

Operating System: Mac OS, Windows

R studio (V 1.0.143)

Cran R (V 3.3.3)

6.5 Hardware Requirements:

System : Intel core i5 processor

Ram : 8Gb

7. TESTING AND RESULTS

7.1 Introduction

Testing is the process of finding difference between the expected behavior specified by system models and the observed behavior of the system. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

7.2 Testing Methods

1) To ensure that during operation the system will perform as per specification.
2) To make sure that system meets the user requirements during operation
3) To make sure that during the operation, incorrect input, processing and output will be detected
4) To see that when correct inputs are fed to the system the outputs are correct
5) To verify that the controls incorporated in the same system as intended
6) Testing is a process of executing a program with the intent of finding an error

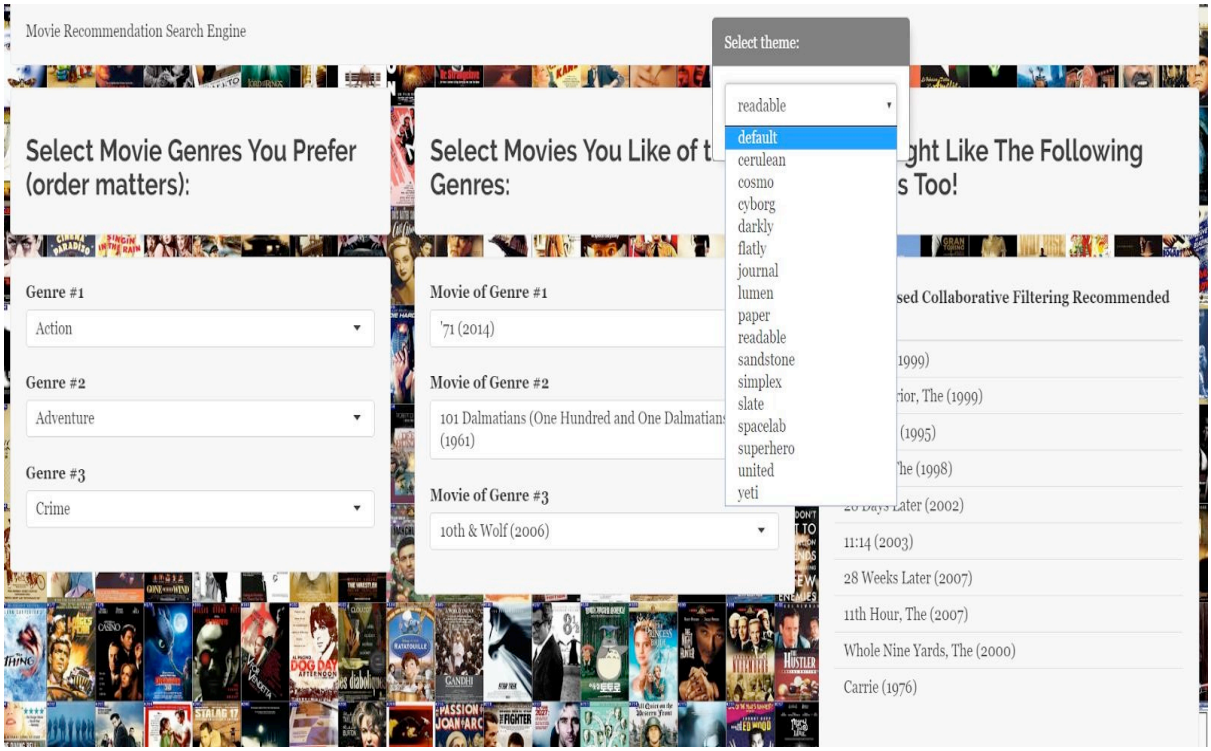
7) A good test case is one that has a high probability of finding a yet undiscovered error The software developed has been tested successfully using black box testing strategy and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers a yet undiscovered error.

Black box testing strategy is a case design method that uses the control structure of the Transaction data set and k out of N strategy is used and designed to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity.

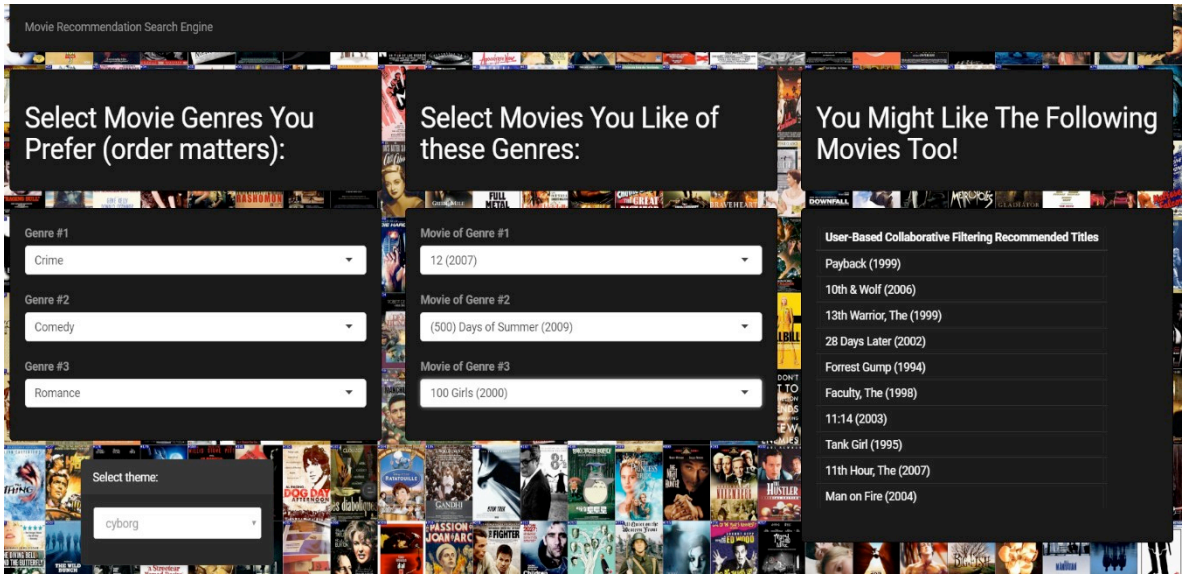
In this application, it does not watch the internal variables during testing. This gives clear idea about what is going on during execution. The points at which the bug occurs were all clear and were removed.

Note that the result of the system testing will prove that the system is working correctly. It will give confidence to system designer, users of the system; prevent frustration during implementation process etc.

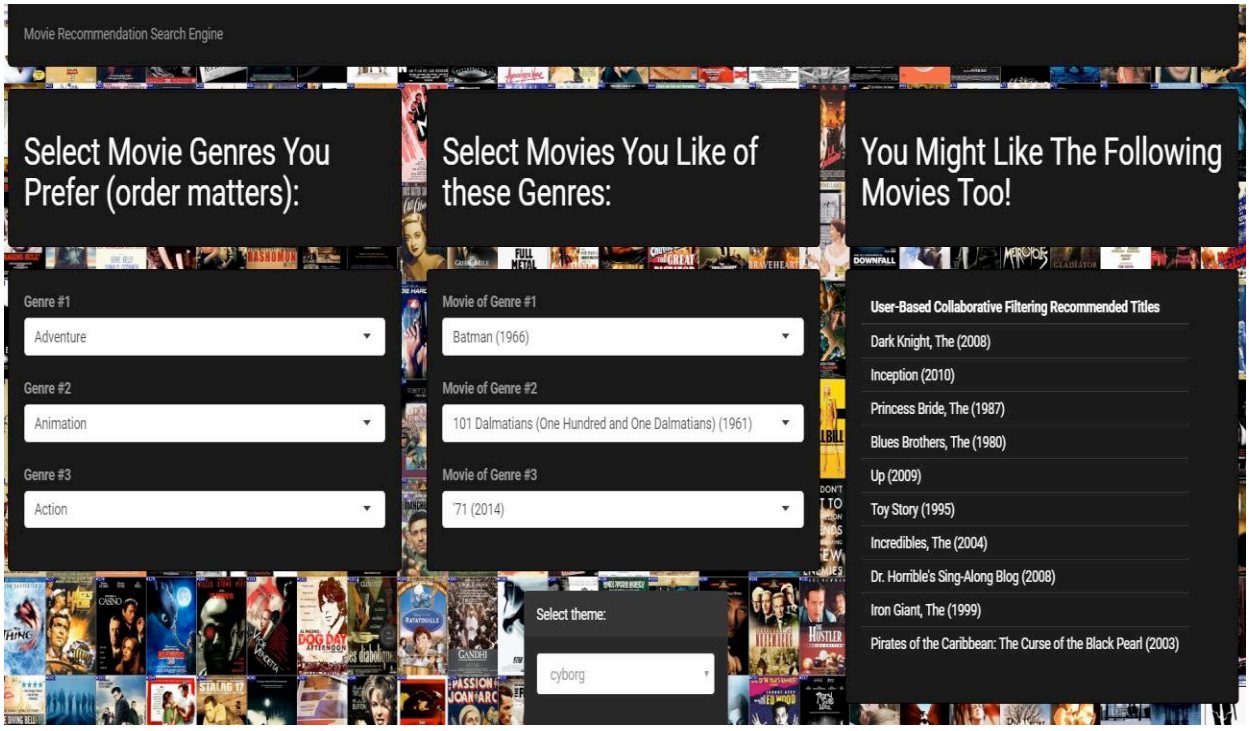
7.3 Results



User has selected Genre Crime,Comedy and Romance and movies that are displayed by default are as below



User had made changed to the movies displayed by default and this has resulted in the updated movie Recommender list.



8. CONCLUSION AND FUTURE WORK

The movie recommendation search engine displays the result list of 10 movies which users may also be interested in. Item based collaborative filtering had less error than user based collaborative filtering. In addition, its less-dynamic model was computed less often and stored in a smaller matrix, so item- based system performance was better than user based systems. Soon, we would like to improve the accuracy of prediction by using Slope One algorithm. Slope One is a family of algorithms used for collaborative filtering, introduced in a 2005 paper by Daniel Lemire and Anna Maclachlan. Arguably, it is the simplest form of non-trivial item-based collaborative filtering based on ratings. Their simplicity makes it especially easy to implement them efficiently while their accuracy is often on par with more complicated and computationally expensive algorithms. They have also been used as building blocks to improve other algorithms. They are part of major open-source libraries such as Apache Mahout and Easyrec.

9. REFERENCES

https://rpubs.com/tarashnot/recommender_comparison
https://link.springer.com/article/10.1007/s10479-016-2367-1#Sec15
https://shiny.rstudio.com/tutorial/
https://muffynomster.wordpress.com/2015/06/07/building-a-movie-recommendation-engine-with-r/