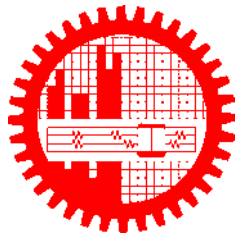


M.Sc. Engg. (CSE) Thesis

# **Handling Long Tail in Recommendation Systems for Niche Markets Using Fuzzy Classification**

Submitted by  
Farzana Hoque  
0413052095

Supervised by  
Dr. Mahmuda Naznin



Submitted to  
**Department of Computer Science and Engineering**  
**Bangladesh University of Engineering and Technology**  
Dhaka, Bangladesh

in partial fulfillment of the requirements for the degree of  
Master of Science in Computer Science and Engineering

September 2019

## **Candidate's Declaration**

I, do, hereby, certify that the work presented in this thesis, titled, “Handling Long Tail in Recommendation Systems for Niche Markets Using Fuzzy Classification”, is the outcome of the investigation and research carried out by me under the supervision of Dr. Mahmuda Naznin, Professor, Department of CSE, BUET.

I also declare that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

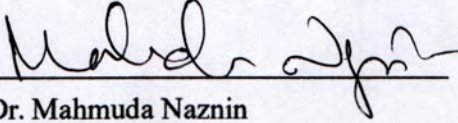
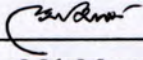
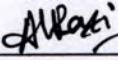
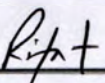
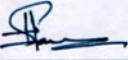
---

Farzana Hoque

0413052095

The thesis titled “**Handling Long Tail in Recommendation Systems for Niche Markets Using Fuzzy Classification**”, submitted by Farzana Hoque, Student ID 0413052095, Session April 2013, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents on September 30, 2019.

### Board of Examiners

1.   
Dr. Mahmuda Naznin  
Professor  
Department of CSE, BUET, Dhaka  
Chairman  
(Supervisor)
2.   
Dr. Md. Mostofa Akbar  
Professor and Head  
Department of CSE, BUET, Dhaka  
Member  
(Ex-Officio)
3.   
Dr. A. B. M. Alim Al Islam  
Associate Professor  
Department of CSE, BUET, Dhaka  
Member
4.   
Dr. Rifat Shahriyar  
Associate Professor  
Department of CSE, BUET, Dhaka  
Member
5.   
Dr. Hasan Sarwar  
Professor  
Department of CSE  
United International University  
Dhaka  
Member  
(External)

## Acknowledgement

I would like to express my sincere gratitude and profound indebtedness to my supervisor Dr. Mahmuda Naznin for her constant guidance, insightful advice, helpful criticism, valuable suggestions, commendable support, and endless patience towards the completion of this thesis. I feel very proud to have worked with her. Without her inspiring enthusiasm and encouragement, this work could not have been completed.

I must express my very profound gratitude to my parents and my spouse for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. Thesis work is always frustrating and challenging but their supports made it easier to me. This accomplishment would not have been possible without them.

Last, but by no means least, I thank Allah for the talents and abilities I was given that made it possible to undertake this research.

Thank you.

Dhaka  
September 30,  
2019

Farzana Hoque  
0413052095

# Contents

<b>Candidate’s Declaration</b>	<b>i</b>
<b>Board of Examiners</b>	<b>ii</b>
<b>Acknowledgement</b>	<b>iii</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Algorithms</b>	<b>ix</b>
<b>Abstract</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Classifications of Recommendation System . . . . .	1
1.1.1 Content Based Filtering . . . . .	2
1.1.2 Collaborative Filtering . . . . .	3
1.1.3 Knowledge-based Filtering . . . . .	5
1.2 Long Tail and Niche Market . . . . .	6
1.2.1 Long Tail . . . . .	6
1.2.2 Niche Market . . . . .	6
1.2.3 Challenges of Long Tail and Niche Market . . . . .	7
1.3 Organization of the Thesis . . . . .	8
<b>2 Related Work</b>	<b>9</b>
<b>3 Problem Domain</b>	<b>13</b>
3.1 Fuzzy Set and Membership Function . . . . .	13
3.2 Preliminaries . . . . .	14
3.2.1 Selection of Item Attribute . . . . .	16
3.2.2 User Profile Generation . . . . .	17
3.2.3 User Choice List . . . . .	18
3.3 Item Profile Generation . . . . .	20

3.4	Recommendation Methodology . . . . .	23
3.5	Data Collection . . . . .	24
<b>4</b>	<b>Results and Analysis</b>	<b>28</b>
4.0.1	Impact of Attribute Quantification . . . . .	30
4.0.2	Impact on the Tail . . . . .	32
4.0.3	User and Movie Class . . . . .	33
4.1	Impact on Attribute Coverage . . . . .	36
<b>5</b>	<b>Conclusions</b>	<b>39</b>
	<b>References</b>	<b>40</b>
<b>A</b>	<b>Codes</b>	<b>42</b>
A.1	SQL Query Code 1 . . . . .	42
A.2	SQL Query Code 2 . . . . .	50

# List of Figures

1.1	Classification of Recommendation System [1]	2
1.2	Content-based and Collaborative filtering technique [1]	3
1.3	User-based Collaborative filtering technique [1]	4
1.4	Item-based Collaborative filtering technique [1]	5
1.5	Knowledge-based filtering technique [1]	5
1.6	Long tail	6
2.1	User-Item rating matrix using bipartite graph.	10
2.2	Fuzzy Inference System Model.	11
3.1	Graphical representation of fuzzy set.	13
3.2	Movie rating decides the fate of movie.	15
3.3	Low rated movies are on the tail.	15
3.4	Attribute of movie.	17
3.5	Proposed quantified attributes of movie.	17
3.6	User profile design approach.	18
3.7	User choice's fuzzy fpecification.	19
3.8	Item Profile Design Approach	21
3.9	Item Attribute's fuzzy Specification	22
3.10	User choice's to movie with fuzzy classification	24
3.11	User choice class representation and the long tail.	26
3.12	Movie attribute class representation and the long tail.	27
4.1	Recommended movie for single user.	29
4.2	Single movie with attribute match for single user.	30
4.3	Recommended to User <i>Robin</i> followed by single rating.	33
4.4	Recommended to User <i>Robin</i> followed by attribute rating.	33
4.5	Attribute class of the user.	34
4.6	Attribute class of the user.	34
4.7	Precision value for proposed method	35
4.8	Recall value for the proposed method.	36
4.9	F-measure value for proposed method	37

4.10 Attribute quantification of different users. . . . .	37
4.11 Attribute quantification of different movies. . . . .	38



# List of Tables

3.1	Table of rating classification. . . . .	14
3.2	Table of User Item rating matrix . . . . .	25
4.1	Standard deviation . . . . .	29
4.2	Standard Deviation for 'Movielence' data-set . . . . .	29
4.3	Table of degree of attribute for a single movie <i>Paltan</i> . . . . .	31
4.4	Attribute Quantification . . . . .	31
4.5	Table of Performance matrix of proposed method . . . . .	35

# List of Algorithms

1	Attribute Rating Classification . . . . .	17
2	User rating classification for attribute . . . . .	19
3	Movie Rating Classification for Attribute . . . . .	21

## Abstract

Recommendation systems provide the best-desired items to a user based on the user's interest measured from user activities. Online movie search is a very popular activity of the internet users. There are many systems, which are suggestive for searching the movie based on the fact that users previously have viewed any other similar types of movies. Items or movies are selected based on their features. However, all of the attributes of the items are not equally important. All popular items are not liked by all users and niche items are available to the classified users. Therefore, the degree choice by the users based on the attributes of items of the niche markets should be considered. In this work, the fuzzy rating has been used for a niche market's items. In this research, we use *Mamdani Rule-Based Fuzzy Inference Technique* for movie recommendation. This methodology helps the movie viewers to watch a movie after knowing that how much utility the user will get from this movie. We find that, proposed fuzzy rating reduces the long tail of the movie list. This helps users to consume a specific item. The movies will be recommended to the specific users because segmentation and the fuzzy membership of the attribute helps to achieve the popularity in niche market. We design a data-set, where the single movie get multiple ratings for different attributes. We find our proposed model can handle long tail problem of the items of niche markets.

# Chapter 1

## Introduction

*Recommendation System* is information filtering system which generates meaningful recommendations to a collection of users for items or products those might interest them. For example, Amazon suggests different books to its users, Netflix suggests popular movies while searching. There are many real world examples of industry-strength recommendation systems [2]. In the era of high internet usage, the recommendation system develops a notion of affinity between users and items, which can be used to identify well-matched pairs and the contents are such as movies, books, magazines, jobs, news, research articles, restaurants, historic places, grocery stores etc. [3]. Because of the rapid growth in the amount of available digital information about products in internet, search engines work as information retrieval system but they do not provide prioritization and personalization of information. Recommendation System solves these issues as information retrieval system more than ever before.

In a conventional system, like a book store, a CD/DVD store, newspaper, etc. have limited physical storage to keep all collections instead of only popular items. Behind this, *niche items* remain mostly away from users and tradings, whereas these items have the potentiality to meet user satisfaction and can aggregate the total revenue of the business. It also abide by *Pareto Principle*, better known as the 80/20 rule [4]. According to this principal *long tail* makes a different view of marketing and niche products and he become apt for profitable market. With the aspects of recent life, online movie platforms has become a popular source of entertainment. Knowledge and the recommendation system is a very effective application for this. The recommendation system gives the users a proper suggestion when they require a movie.

### 1.1 Classifications of Recommendation System

In this section, the classification of recommendation system is described. Figure 1.1 is a complete basic classification of recommendation system. This techniques are used on different applications

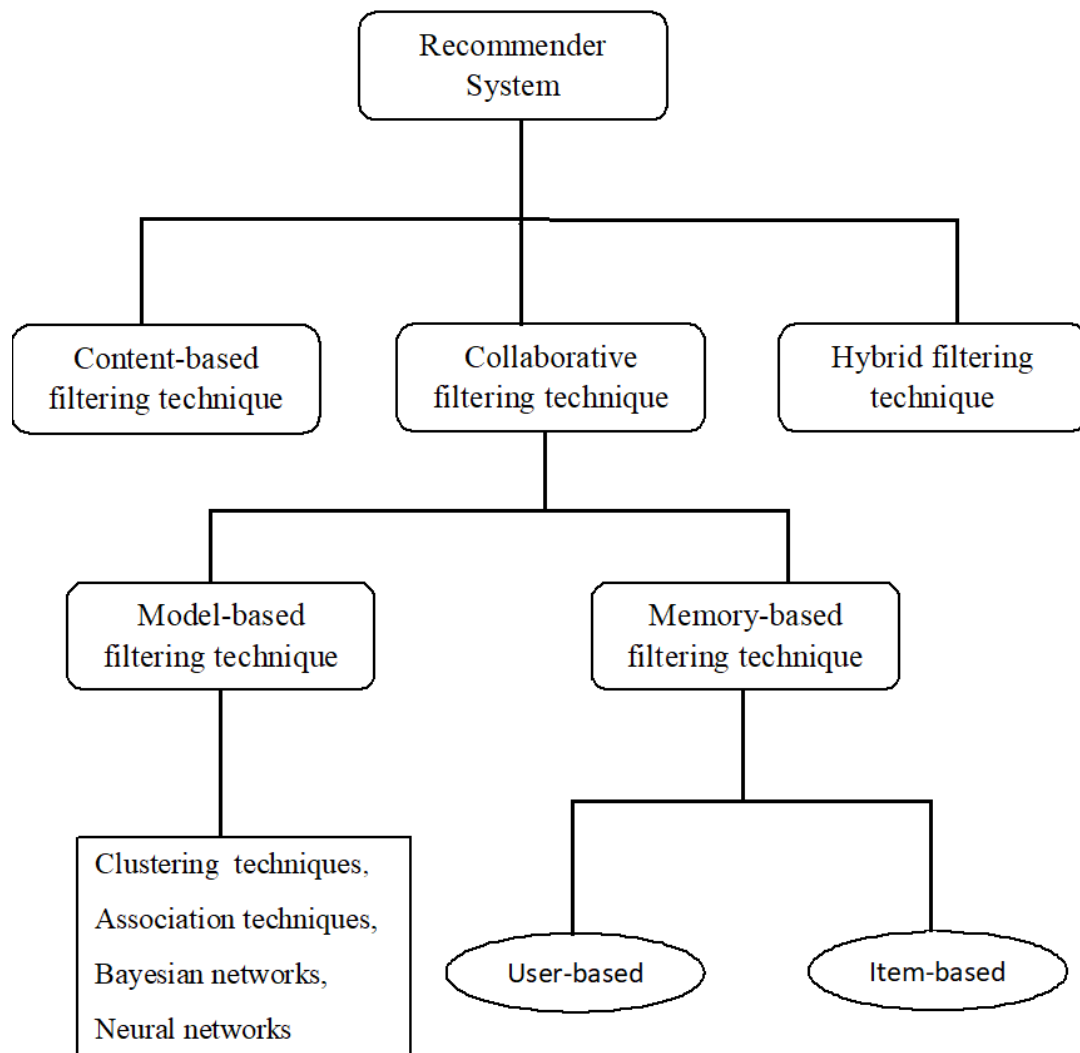


Figure 1.1: Classification of Recommendation System [1]

as the need of application plot. Some of this techniques are described here.

### 1.1.1 Content Based Filtering

*Content-based Filtering* is a recommendation technique which considers item properties for recommendation. Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions [1]. For example, web pages, publications, news, research-articles, e-learning, etc. can be recommended using this technique. Content-based filtering use domain dependent algorithms and there is no dependence on users. It uses *Vector Space Model* such as *Term Frequency Inverse Document Frequency (TF/IDF)* or probabilistic models such as *Naïve Bayes Classifier*, *Decision Tree* or *Neural Network* to model the relationship between different documents within a corpus [1]. Figure 1.2,

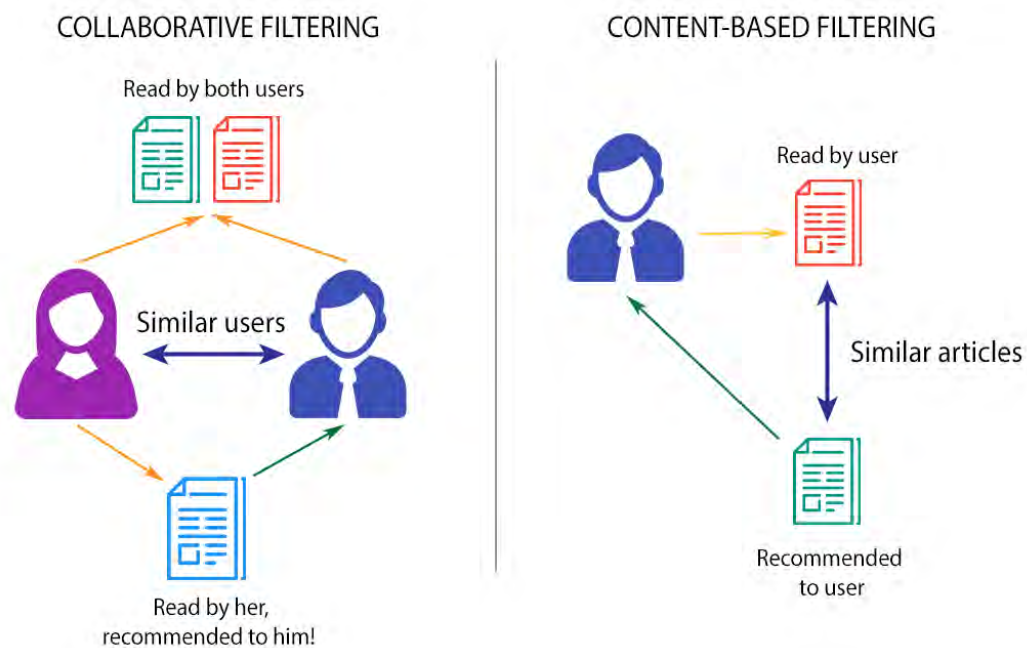


Figure 1.2: Content-based and Collaborative filtering technique [1]

is the demographic example of content-based filtering where similar item is recommended to users.

## 1.1.2 Collaborative Filtering

This recommendation technique recommends users using features extracted from the users' past recommendation. Collaborative filtering technique is a domain-independent technique which is applicable for content that cannot easily and adequately be described by metadata such as movies, music, etc. It works on user preferences to items as making user-item matrix which is used to calculate similarity and predict ratings from user's behaviour and recommend the similar item to neighbourhood users. Figure 1.2 is an example of collaborative filtering where, if one item is consumed by a user, then it can be predict that item will be liked by the user's neighbourhood users. Collaborative filtering technique is divided in two categories: Model-based and Memory-based filtering technique. Memory-based filtering technique works on two categories- *User-based Memory Filtering* and *Item-based Memory Filtering*.

### Model-based Filtering

*Model-based* filtering technique considers the user's previous ratings from user-item matrix to learn different models and to compute the similarity to predict the ratings for neighbourhood users to recommend. Examples of this technique's model are including Dimensionality Reduction such as Singular Value Decomposition (SVD). Matrix Completion Technique, Latent Semantic

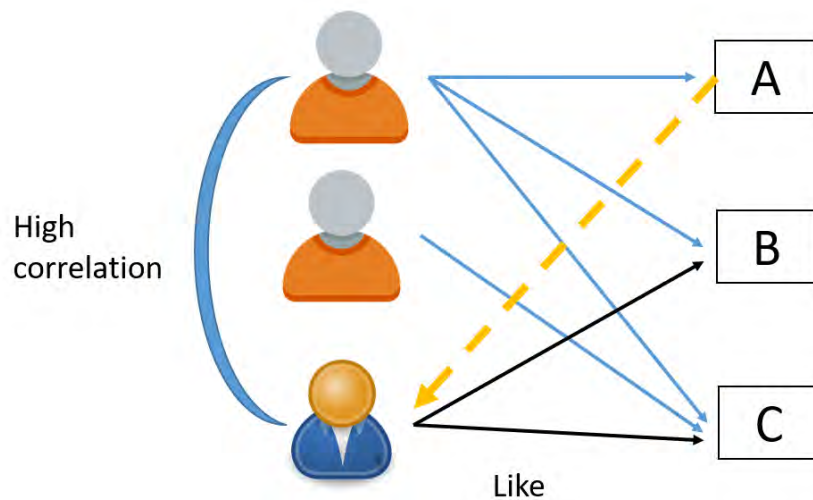


Figure 1.3: User-based Collaborative filtering technique [1]

methods, and Regression and Clustering. Model-based techniques analyze the user-item matrix to identify relations between items; they use these relations to compare the list of top-N recommendations [1]

### User-based Filtering

*User based Collaborative Filtering* calculates similarity between users by comparing their ratings on the same item, and it then computes the predicted rating for an item by the active user. This is context independent and compare to other such as content-based technique it is more accurate. But it has the cold start problem with new users, who will have no to little information about them to be compared with other users. Also the percentage of people who rate items is really low. Figure 1.3 is graphical example of user-based collaborative filtering technique.

### Item-based Filtering

*Item-based collaborative filtering* technique computes similarity between item by comparing their ratings on that or similar item from neighbourhood user, not the similarity between two user. There are some similarity measurement techniques to compute similarity between two rating set, count as vector of item. These are Cosine-Based Similarity, Correlation-Based Similarity, Adjusted Cosine Similarity, Jaccard distance, etc. used for similarity computation. Figure 1.4 is an example of item-based collaborative filtering where item- item correlation has been prioritized.

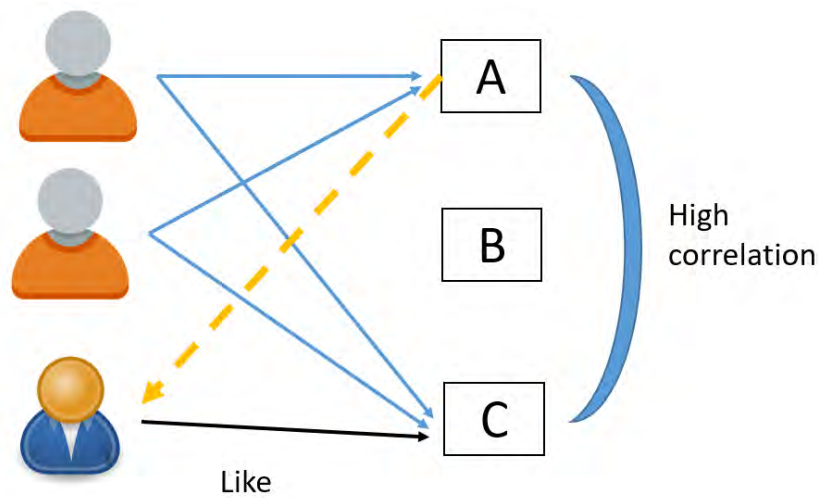


Figure 1.4: Item-based Collaborative filtering technique [1]

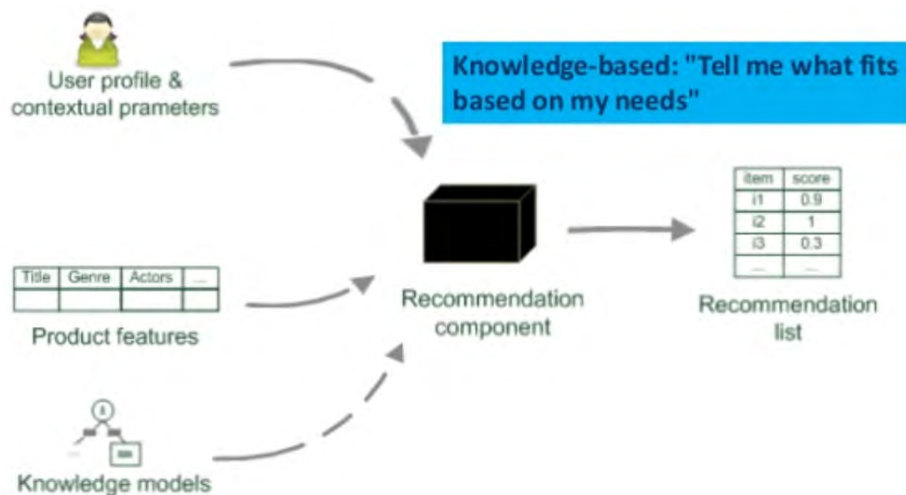


Figure 1.5: Knowledge-based filtering technique [1]

### 1.1.3 Knowledge-based Filtering

Knowledge-based filtering technique is combination of Content-based and Collaborative filtering technique to overcome their limitations and problems and build an unified algorithm to provide an effective and more accurate recommendation. It uses different model of different technique and approaches combined multiple algorithm, if one algorithm has limitation other will solve that issue. This recommendation is used applied for apartment booking, car sell, and etc. where user profile and item profile both need to be considered for the best item to an appropriate user using a huge amount of data analysis for the recommendation. Figure 1.5 is an example of structured model of knowledge-based filtering technique.



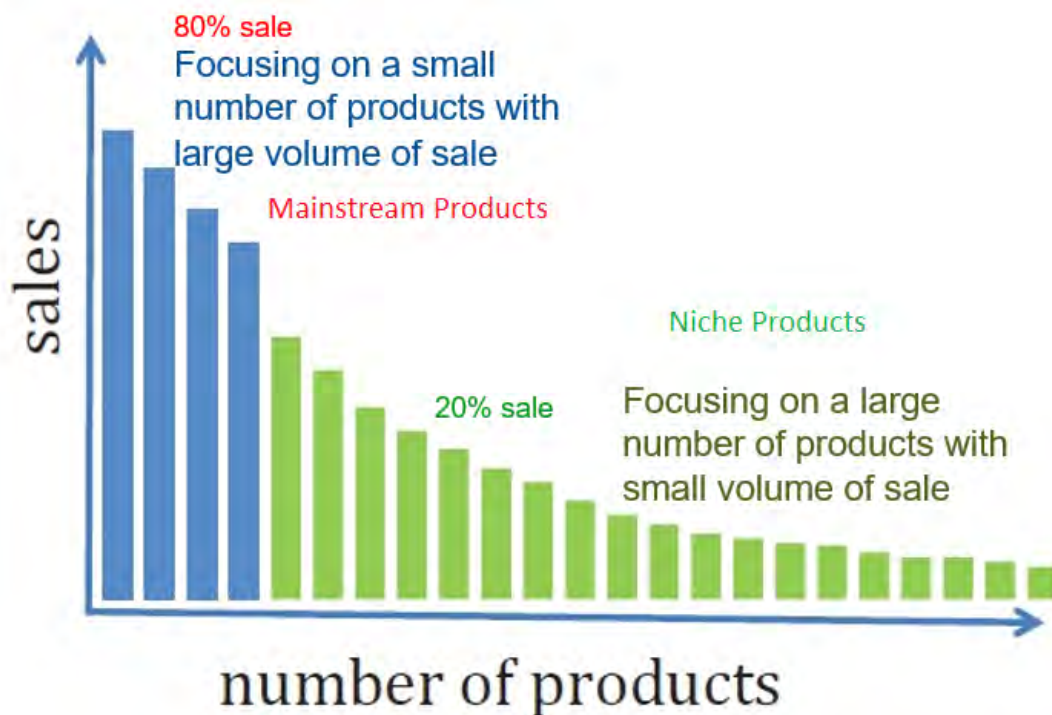


Figure 1.6: Long tail

## 1.2 Long Tail and Niche Market

In this section we discuss the different challenges of *long tail* and *niche market*.

### 1.2.1 Long Tail

*Long Tail* is created due to the fact that the economy is increasingly shifting away from a focus on a relatively small number of mainstream products and markets at the head of the demand curve to toward a huge number of *niches* in the tail. As the costs of production and distribution fall, all products do not get focus of consumers. Moreover, due to the limitation of space and resources many products do not come to the mainstream store in conventional market place. *Long Tail* concept abide 80/20 Perito Principle [4]) rule. It shows the most popular products lie with the less popular items in aspects of business credit. According to the traditional business, small number of products are sold rapidly, and the large number of products stay with less sale. Figure 1.6 describes the scenario of long tail. The long tail is used to describe the phenomenon that less sale products can be the future business [5].

### 1.2.2 Niche Market

*Niche market* is a part of mainstream market where niche items have specific unique features for satisfying specific customers. This market is a specialized market which is limited and has

clearly defined a range of products for a specific group of customers. Every product can be defined as a market niche. It is marketing to a more narrowly defined consumer group which seeks products or services tailored specially for the individual needs and preferences. Niche marketing is very common on the Internet's, on various websites [6]. In traditional market, there is scarcity of space and resources. But in Internet world, there is abundance of space and resources. For example, Amazon or any other online platform's have limitation of storage and became 'one-stop shopping convenience' which aggregate the total revenue of business and user satisfaction [5, 7].

Niche market has lot of advantages to make profit using customer need and personalization of products. It has low competitors in market and to earn best profit need to do best research on customer need and mainstream market analysis. It is important to implement an efficient niche marketing strategy for the establishment of this market.

There are two important facts of niche market is market segmentation strategy and positioning of niche products. In [6], the authors proposed a topic-based *Hierarchical Bayesian* model for niche product recommendation. They only use the user-item ratings matrix and movies' descriptive content information. They did not classify or distinguish popular item, niche item, and long tail item.

A *market segment* is a segmented category of customers who have similar likes and dislikes in an homogeneous market. These customers can be individuals, families, businesses, organizations or a blend of multiple types. Market segments are known to respond somewhat predictably to a marketing strategy, plan or promotion. It employs a strategy to differentiated marketing approaches for subset of a market with unique needs and preferences. Segmentation of a markets can be developed in several ways, such as

- Geographically by region or area,
- Demographically by age, gender, family size, income, or life cycle,
- Psycho-graphically by social class, lifestyle, or personality,
- Behaviorally by benefit, use, or response.

### 1.2.3 Challenges of Long Tail and Niche Market

There are limitations of long tail and niche market with a lot of opportunities. Basically, the tail items of the long tail from mainstream market goes on niche market with lot of risk. These items face challenges of niche markets which are are discussed as follows. To get profit need to understand the customer need and satisfaction. Although this is cheaper than mainstream market, but there is high risk about to gain the customer satisfaction and get profit. Niche products are not be aggregated to the business all the time. For example, if we consider the online movie platform,

some movies really go on bottom because of the missing of rating or marketing. Moreover, some movie items comes on the top of sale which are not most liked by the audience and if one movie is picked up from the tail or low rated, it would be just time wasting for audience. And niche products are limited to serve with minimum number of customer, that's a big problems of niche market.

We now provide the main contribution of the thesis. *Positioning* is the practice of designing a product, item or brand to have a unique selling proposition relative to the competition, aspects of users. This can include factors such as product, price, distribution, promotion and brand identity etc. In this thesis, positioning in long tail is be factorized with attribute, item rating of the membership class which it belongs to. Then positioning the items will be recommended to the users.

### **1.3 Organization of the Thesis**

In this section, we provide the organization of the thesis. The rest of this thesis is organized as follows. The review of the related work is in Chapter 2. In Chapter 3, the problem formulation is elaborately described. The experimental results and comparisons are reported in Chapter 4. Finally, Chapter 5 provides conclusions.

# Chapter 2

## Related Work

In this chapter, we discuss some relevant recommendation systems.

Now in these internet times, peoples are going online for even a single thing to buy or choose like: electronics gadgets, books, movies, clothes, household things, research articles, restaurants, historic places, shopping malls, twitter pages, and many more products as their needs. First of all, the users look for these in a specific web site or search engine. Search engine solves the information overload but they do not provide personalization of data [2]. Many work have been done on this topic, still, it is a very challenging area because of the diversity under the domain of data science.

There are so many applications of recommendation system which are always companion in our daily life [2]. Along with the basic three recommendation systems, [7] describes the challenges and benefits of *Demographic* technique based and *Keyword* based recommendation system. In demographic recommendation system the user's characteristics such as gender, age, education, etc. are considered and the user rating or reviews are not considered.

On the other hand, the keyword-based recommendation system makes user preferences from the keyword of the user's text reviews. In [2], the authors focused on the electronics gadget as product and reviews as feedback. They used *Collaborative Filtering* recommendation system with the keyword-based technique and developed a recommendation system with classified users like "admin user" and "active user", where Admin user can log in through the interface and upload reviews, add products, perform training on the data. And an active user also can log in, select a product, provide reviews for the product, view his reviews and when he clicks on the recommend link gets the recommended products. But there is no activity with "rating" or any product specified recommendation, sometimes the only keyword cannot identify the diversification of the product and the niche products are not focused properly in this system.

As compared with the conventional and digital market, a seller can only keep the "popular" or "must be sold out" type items with a large volume because of the scarcity of space and resources in the conventional market. On the other hand, in the online digital market comes with an abundance of space and resources. This state of market long-tail concept supports "one-stop

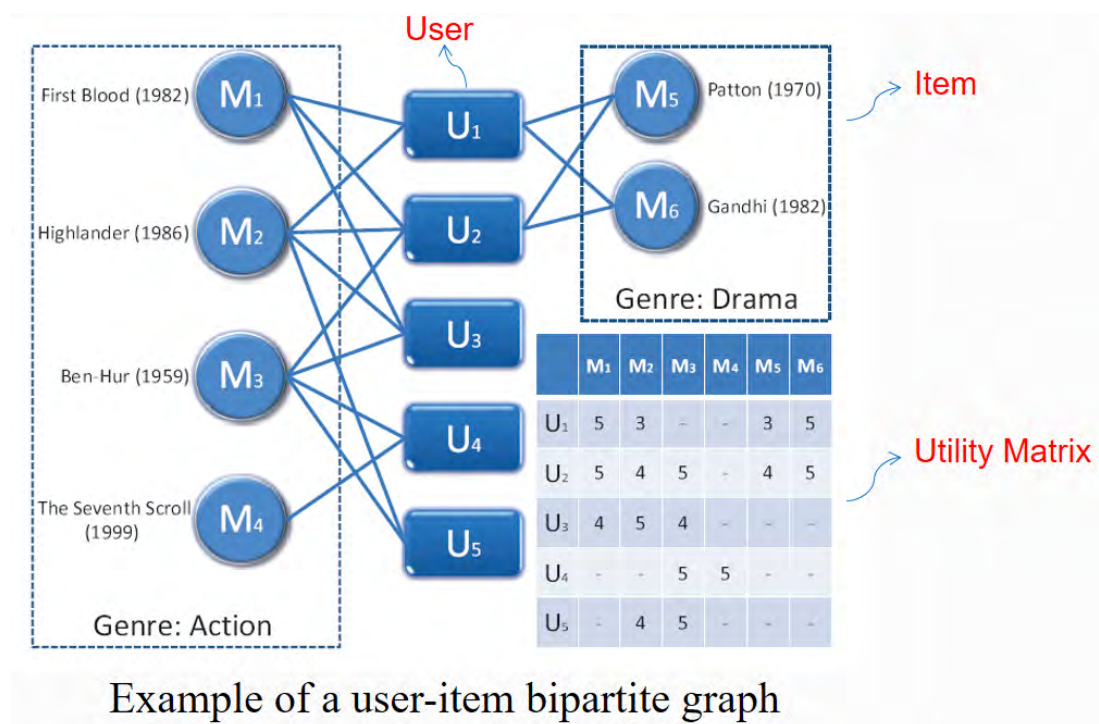


Figure 2.1: User-Item rating matrix using bipartite graph.

shopping” which provides mainstream and niche products. In [7], they proposed a graph-based algorithms for the long tail recommendation where user-item information represented in an undirected edge-weighted graph representing in figure 2.1, and analyzed theoretically the foundation of applying Hitting Time algorithm with considering the random walk similarity for long-tail item recommendation aspects of collaborative filtering recommendation system. From their proposed hitting time algorithm they recommend the item with minimum hitting time to similar user which differ from traditional collaborative recommendation. But the item with minimum hitting time, has most probability to be inappropriate for the average user. They used user ratings on the item and then applied absorbing time and cost with user entropy where there is a chance to fall into the niche market. Because all time user will not prefer the tailed or less selling products. Also, they were not focused on the item’s category diversification at all.

On the other hand [8], the authors carried out detailed feature engineering work and proposed a new evaluation criteria system that improves the long tail recommendation rate and stability. They consider user behavior, product information and all features which are categorized into three categories. The first category is the information of user characteristics, second is the information of behavior characteristics and third is the information of the platform. They build feature’s tree, using the cascade model and classified the item in proportion to 8:2 where the smaller part is from long tail, which sometimes distorts the niche market.

In [9], using the fuzzy logic concept for user preferences the author builds a shortlist of audio gadget products to recommend which is likely to match the *Pereto-Principal* rule and their

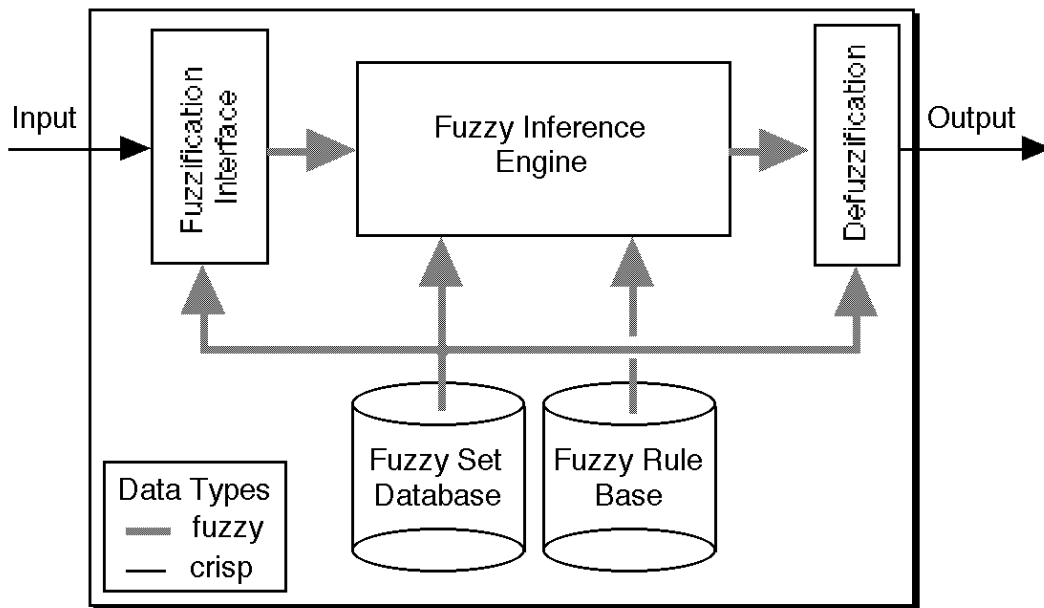


Figure 2.2: Fuzzy Inference System Model.

system only recommends top listed items. They consider budget, reviews, genres, sound quality, etc as a fuzzy inference system's (figure 2.2) parameter and apply rule-based *Mamdani Fuzzy Model* for each parameter. Where this system provides user weight-based adaptation for best budget-friendly recommendations. But this is not applicable for niche items such as movies or books where the category has many diversification and choices vary for different users vastly.

In [10], algorithms based on fuzzy concept like fuzzy c-means, k-means clustering have been used in machine learning-based system and also used for rank prediction for scientific research papers. Here publication date, authors, keywords, citations indicate the rank of these papers. Those are classified items. They also developed their algorithm focused on research paper article as item which is appropriate for higher studies education area. However it does not have any solution for long tail handling.

In the case of item-oriented markets like jobs, books, movies, songs, etc where user and item are directly personalized and there is a one to one relation between user and item. Sometimes it represents using the user-item matrix with their dimension. In [11], they build a low dimensional subspace of a novel inter-item proximity matrix which consisting of a similarity and a scaling component of user and item. They proposed the *EIGENREC* algorithm based on a *pure-SVD* (Singular Value Decomposition) algorithm where *pure-SVD* is popular for matrix factorization. But they cover the collaborative recommendation where item attribute's rating is not considered.

Considering the average scenario of recommendation systems, the users have the full consent to

give any rating on product feedback. But all users would not be good decision-makers all the time in give rating. It depends on the knowledge of the users. As a result, some anonymous or noisy data will be added to a good product. To resolve this problem [12] works on this issue using a fuzzy set classifier to remove noisy data as divided them in *strong*, *weak*, *average* class and used *Bhattacharya* coefficient and *Pearson* coefficient similarity for noise-free rating for recommending to the user [12]. However, their work is good for a single rating in one product aspect of a collaborative filtering recommendation system but not suitable for category wise rating classification. They also show that the performance matrix worked well with the existing method.

In business markets, all users do not behave in the same manner [13]. Therefore, recommendation models substantially vary in the markets. Besides, to predict every user's potential rating for a specific item, existence of only one score for an item is generally used by currently available models such as fuzzy tree structure [14], bi-clustering and fusion methods [15], fuzzy c-means [10], etc. These models do not consider item's degree of attributes, i.e., how precisely an attribute can add value to the item is also important. Other recent studies [12], [9], also do not consider this either. Thus, consideration of degree of attributes, especially for the niche markets an item is yet to be considered.



# Chapter 3

## Problem Domain

In this chapter, we formulate the problem and describe our solution approach.

### 3.1 Fuzzy Set and Membership Function

*Fuzzy Set:* A fuzzy set is a set of real numbers  $x_i$  which represents the membership values of  $u_i$  which (generally) lies in the range from 0 to 1. A fuzzy set is represented by a set of pair  $u_i / x_i$ , where  $u_i$ , is the membership value of  $x_i$  [16]. For example, let us consider  $X = g_1, g_2, g_3, g_4$  be the reference set of students and  $A$  is the fuzzy set of *smart* students, where *smartness* is a fuzzy variable for the students with value pair set  $A = (g_1, 0.8), (g_2, 0.6), (g_3, 0.7), (g_4, 1)$  Here,  $A$  indicates the *smartness* value of  $g_1$  is 0.8. Figure 3.1 explains the example.

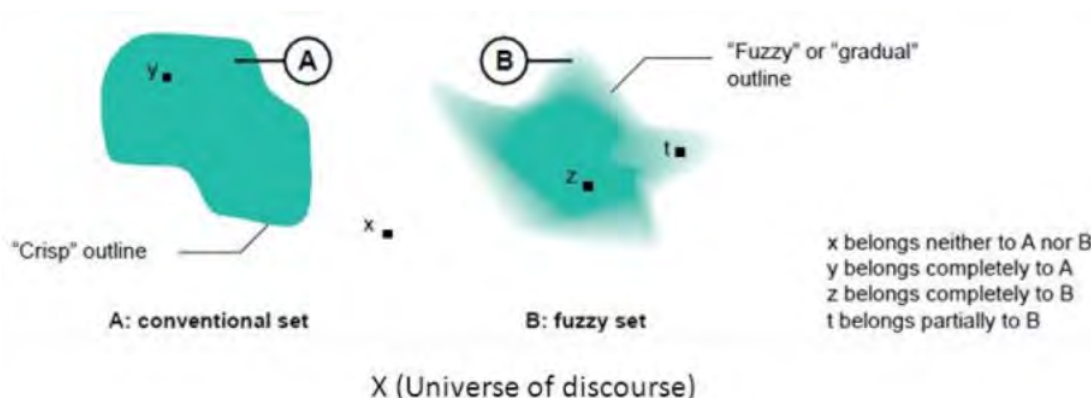


Figure 3.1: Graphical representation of fuzzy set.

*Fuzzy Logic* is an approach to compute based on degrees of truth rather than *boolean* value for *True* or *False* (1 or 0). Figure 3.1 helps to understand the representation of the fuzzy set and its membership value.

In our model, we use fuzzy set for classifying the rating. *Rating* is considered as the degree of membership. We consider the rating range from 0 to 5 and put them *poor*, *low*, *medium*, and



*high* class for each attribute. For example, the user's feedback that gives the movie rating value considers user interest also for find the quality of the rating. Therefore, the rating for movie can be weighted.

Table 3.1: Table of rating classification.

Rating	Class
0.1-1.5	Poor
1.6-2.5	Low
2.6-3.5	Medium
3.6-5	High

Table 3.1 represents rating category and range. *Poor* has the rating 0.1-1.5, *High* class with rating range 3.5-5.0 defines the degree of the class.

## 3.2 Preliminaries

Online business platforms are more popular for recommendation systems where users do not need to use a search engine to find an item because users automatically get recommended or suggested from the user behavior and profile. This is possible after analyzing the user's profile and behavior pattern. Usually, users give their feedback as a rating, *like*, *dislike* or any other activities such as *watch*, *purchase* or *use*. Any user's feedback can be of two values either yes (1) or no(0). Our proposed system focuses on the feedback with some quality.

For example, an user *A* likes a movie *X-man* and gives a rating 4.5 to 'X-man' can be considered as a movie of *Action*, or *Thriller*, or *Sci-fiction*, etc. On the other side, another movie named *Breathe* can belong to *drama*, *thriller*, *story-line*, etc. However, user *A* gives rating 2.0 for *Action* class. The movie will not sustain as an *Action* movie in the box office because of its poor rating. Many users will not consider this even. These types of movie will position at the in *tail* always as review is poor. Figure 3.2 and 3.3 are the examples of niche item in online platform.

We propose a new classification method based on user's choice using *Fuzzy inference* which maintains the input data's quality [17]. Fuzzy inference technique works using input value and applies a different rules for decision-making problem.

*Mamdani* method is the most commonly to use fuzzy inference technique. *Mamdani Model* is a knowledge-driven predictive model, it works with input of crisp data and also with different intervals or linguistic terms. The major advantage of this model is-it provides a measure of confidence for predicting future value when the actual value is unknown. The important domain of its applications are web-based applications such as online shopping platform, movie site, online book stores, etc. [17].

Therefore, *Fuzzy Mamdani* model is good for our proposed method for movie recommendation. Attributes ratings will be decided based on different levels of membership functions or based

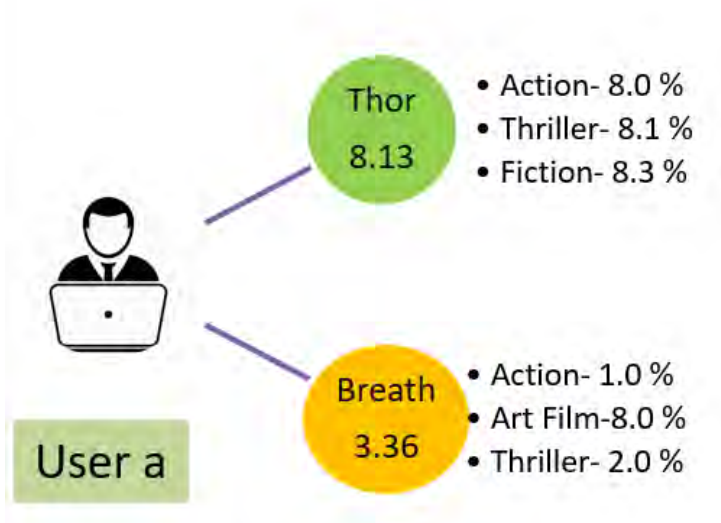


Figure 3.2: Movie rating decides the fate of movie.

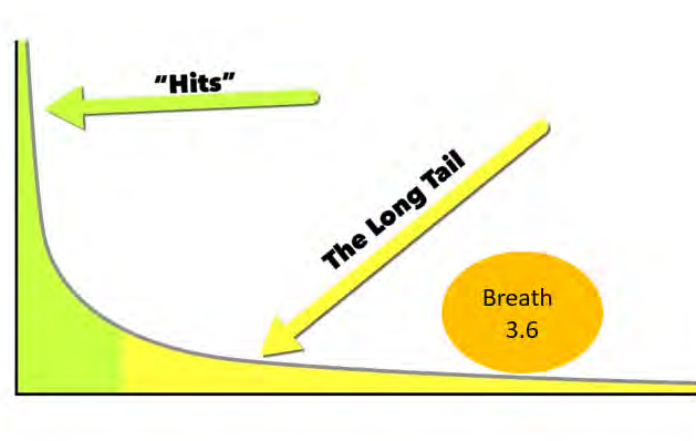


Figure 3.3: Low rated movies are on the tail.

on their degree (existence). In this model, the long tail produced, where users choices can be measured from the user profile. There will be the same classification for user choices on different categories and preferences obtained by fuzzy function.

If movie attributes become more classified and movies are recommended to the right user, an important aggregation in business with maximum user satisfaction will be there. This is also applicable for the interaction perspectives for user choice attributes. *Recommendation System* tries to provide the best-desired items to a user as per the user's interest in measuring activities such as rating, preferences, etc. Capturing the user's activity on an item can provide a better recommendation system. To do this, a new customized configuration for user and item has been proposed here.

In the proposed system, the user provides preferences for the items. Items are classified based on the degree of the choices of the users' based on *fuzzy set theory* where the conventional

recommendation systems are based on the *crisp set*. Fuzzy theory will help to classify and measure the consistency of rating more precisely for both of the users and the movies or the items. This system has the following steps.

- Step 1: We select appropriate attributes for user choice and movie.
- Step 2: We give rating for classification for attribute.
- Step 3: We make fuzzy set based user profile according to their choices.
- Step 4: We make movie profile and its classification based on fuzzy set theory from the given degree of attributes.
- Step 5: We make recommendation for users from highly preferred attribute match with the movies' degree of attribute.

### 3.2.1 Selection of Item Attribute

There are too many attributes in movie and user choice. But some attributes will be compressed with maximum similarity and less difference.

Some attributes can be liked this. Action, Thriller, Adventure, Mystery, Drama, Romance, Horror, Crime, Comedy, Science-Fiction, Documentary, Story Line, Performance, Print Quality, Animation, etc. These attributes also define the user's choice or interest area. Some of them are also known as *genre*. But here some other attributes change the genre. These attributes preference value as feedback from the user is *yes* or *no* but the proposed method offers between *yes or no* as the degree of choices or interest. This will help to predict items attribute and user choices more in-depth.

For example, let us consider the movie *Thor*, which has three(Action, Adventure, Fantasy) attributes. Another user group defines this movie with two (Fantasy, Sci-Fiction) attributes. For this item confusion is created which one would be its attribute.

In our model system, admin or authorized users will be allowed to upload a movie and add specific attributes with consistency range by rating. This attribute list can be changed by the users who will watch and they can add more attributes with a rating. In this way, the movie will have dynamic attribute list and rating. This model will help the user to understand the movie specification precisely.

We find in Figure 3.4, the movie *Thor* has three attributes. These attributes are focused in this movie and the overall average rating is the only way to decide to watch this movie. But in our new system, shown in Figure 3.5, tells us about the maximum coverage of attributes suits with this movie and also with the quantity it contains. It ensures the user about each attributes consistent with the movie and make user satisfaction precisely.

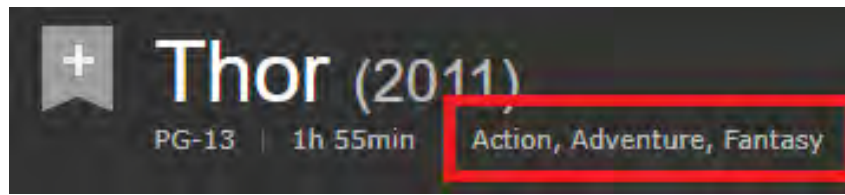


Figure 3.4: Attribute of movie.

Title	Thor
Description	Directed by Kenneth Branagh. Produced by Kevin Feige. Screenplay by Ashley Edward Miller, Zack Stentz, Don Payne
Attributes with RMS ratings	Action : 4.12(High 82.46%)   Adventure : 4.53(High 90.55%)   Fantasy : 4.53(High 90.55%)   Sci-Fi : 4.53(High 90.55%)   Animation : 2(Low 40%)

Figure 3.5: Proposed quantified attributes of movie.

Attribute consistency of users and items will be classified using the algorithm 1. It helps to determine the user choice for a recommendation. The degree of membership for class is quantifying the measurement of user choices and movie attributes.

### 3.2.2 User Profile Generation

In recommendation system, *user profile* is a collection of information and interest area associated with a user. User's basic information such as Name, Address, Gender, Age, Profession, etc. are considered the digital identity of that person. Here, the item is a movie. Which kind of movie he/she prefer to enjoy more or less, and all information will be considered to create the user profile. From different activity such giving rating, views, like, dislike, etc. all are counted as user's activity. In our recommendation methodology, there are two types of user. One is *admin* user and *authorized* users, and the other is *general user*. The first type of users are permitted to upload movie with maximum attribute it contains, and the second type of users are permitted to view, give ratings, the last category users add more attributes with rating. The user will put a

---

#### Algorithm 1 Attribute Rating Classification

---

**Require:** Rating value in attribute

**Ensure:** Degree of membership in fuzzy class and Root Mean Square (RMS) value

**Step 1:** Get the rating value for each attribute as  $R_a$  and calculate  $RMS$  value as crisp input.

**Step 2:** Get the degree of membership class using rating  $R_a$   $RMS$  value as *poor*, *low*, *medium*, *high*.

**Step 3:** Calculate the percentage value from degree of membership value,  $R_a$   $RMS$  for each attribute.

**Step 4:** Prepare the attribute's rating as degree of membership class,  $RMS$  value, and percentage value.

---

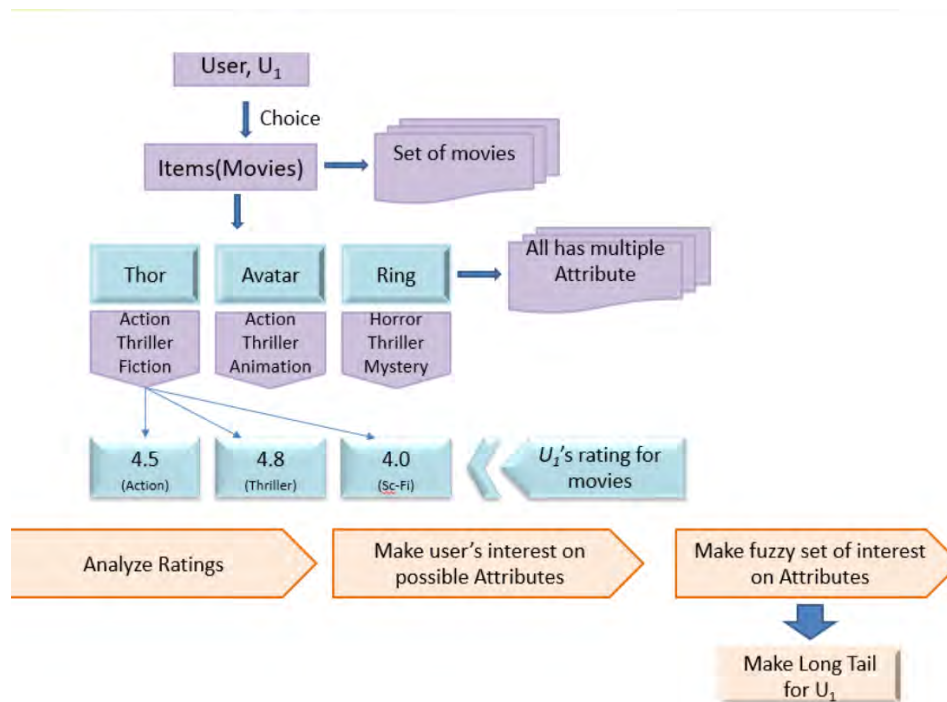


Figure 3.6: User profile design approach.

rating on different attributes of a movie as they enjoy, these will make their profile according to each attribute. These ratings will be used by fuzzy classification for selecting degree of attributes. Figure 3.6 represents the user choice design model. This system will collect all rating of user for different attributes on different movies. Then it calculates the RMS value for each attribute. Afterwards, it converts them into fuzzy set value. This attribute's set value will represent the user choice list.

Figure 3.7 shows the user choice specification using fuzzy set theory and the generation of long tail from each choice weight (ratings). In this system, the single rating has been broken into possible multiple values, which classifies the actual interest for each attribute. It helps to decide item should recommend.

### 3.2.3 User Choice List

The following algorithm 2 classify the user rating which is described bellow.

**Step 1:** For user  $U_a$ , the feedback as rating on different attributes of movie *Thor* are (8.30), *Avatar* (7.17), and *Ring* (7.5).

$$U_{a_{Thor}} = (\text{Action}=3.5, \text{Thriller}=4.0, \text{Sci-Fiction}=4.0)$$

$$U_{a_{Avatar}} = (\text{Action} = 4.0, \text{Fantasy}= 4.5, \text{Thriller}=4.5)$$

$$U_{a_{Ring}} = (\text{Horror}= 4.0, \text{Mystery} = 3.0)$$

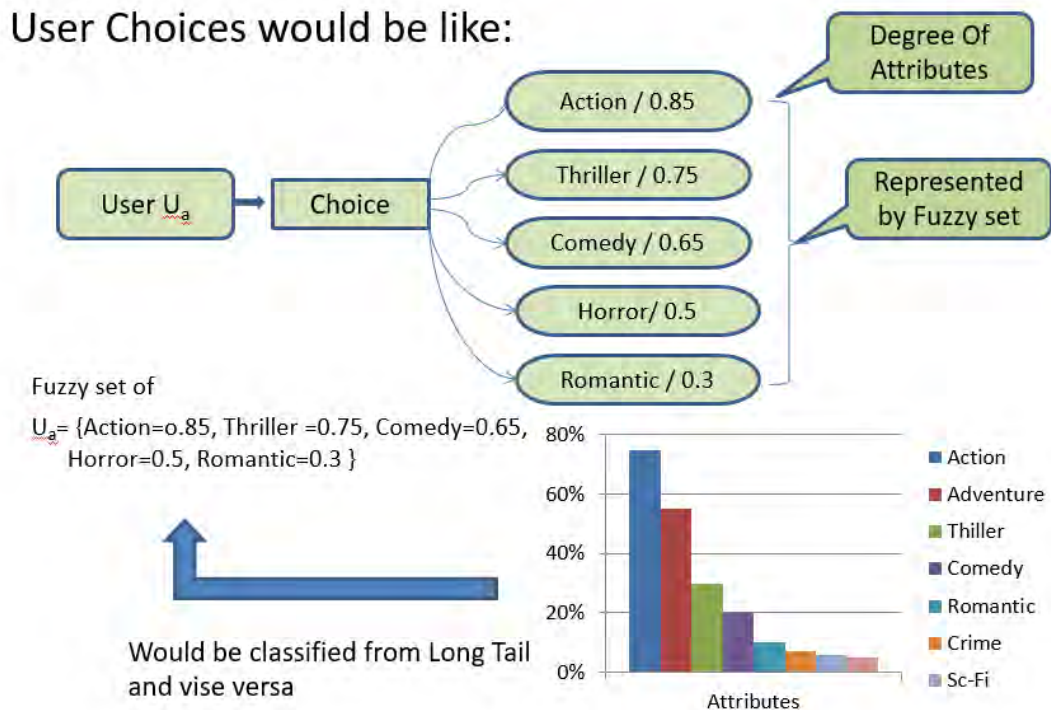


Figure 3.7: User choice's fuzzy specification.

**Step 2:** The classified set of user *Ana* for her attribute wise rating, calculate Root Mean Square value.

$U_{a_{RMS}} = (\text{Action}=3.75, \text{Thriller}=4.25, \text{Mystery}=3.0, \text{Horror}=4.0, \text{Sci-Fiction}=4.0, \text{Fantasy}=4.5)$

**Step 3:** Convert the classified set to fuzzy set of *Ana* using fuzzy set theory.

---

#### Algorithm 2 User rating classification for attribute

---

**Require:** Users rating on movies attribute.

**Ensure:** Fuzzy set of user choice as membership value of the attribute class.

**Step 1:** Get the registered user choice attribute with weight from user profile.

**Step 2:** Optimized all attributes rating value  $R_a$  of user  $U_i$ , and calculate  $RMS$  value as crisp input.

**Step 2:** Calculate the degree of fuzzy membership class as *poor*, *low*, *medium*, *high* for user  $U_i$ .

**Step 3:** Compute the percentage value of membership class of rating  $R_a$   $RMS$  for each attribute.

**Step 4:** Sort the attribute's rating value in descending order and use for create long tail.

**Step 5:** Save the data in three output as membership degree value,  $RMS$  value and percentage value.

---

$U_{fuzzyset} = ( \text{Action}=0.375, \text{Thriller}=0.425, \text{Mystery}=0.30, \text{Horror}=0.40, \text{Sci-Fiction}=0.40, \text{Fantasy}=0.45)$

**Step 4:** Final choice list of user *Ana* is in percentage and membership class.

$U_{choicelist} = \text{Fantasy} \mid 90 \text{ percent} \mid \text{High},$   
 $\text{Sci-Fiction} \mid 80 \text{ percent} \mid \text{High},$   
 $\text{Thriller} \mid 85 \text{ percent} \mid \text{High},$   
 $\text{Horror} \mid 80 \text{ percent} \mid \text{High},$   
 $\text{Action} \mid 75 \text{ percent} \mid \text{High},$   
 $\text{Mystery} \mid 60 \text{ percent} \mid \text{Medium},$

This choice List of user will be used to measure the degree of attribute, match with movie's attribute list for recommendation.

### 3.3 Item Profile Generation

In our 'Movie Recommendation' movies will be uploaded with basic information such as Name, Director, Production, Release Date, Cast, and etc. Some attributes will be added to the movie as core attribute by authentic users. Then general user, after watching the movie he/she will be able to add more attributes and the rating with existing attributes. All rating from users for a single movie will create movie profile. This profile will be used for make recommendation to users according to the degree of attributes.

In general, after enjoying the movie as newly released, recommended or searched, the user will put some rating on a different attributes of the movie. After that, the user just give one rating according to default movie attributes. And also in a different online movie site, movie attribute become changed. In this situation sometimes users choose a movie with their favorite genre (in general), but after sometimes, some movies tagged with action, drama, comedy, etc. genre and people who like action very much, get recommended to them. But after the user watched, he/she realized this movie is more on drama and comedy with less action, and users get disappointed. To solve this issue, admin and user can give their rating in each attribute to express how much they like this movie as in attributes consistency, i.e. how much action is in this movie from user view.

Each movie will be classified by rating due to their attribute consistency and then, there will be a long tail for each attribute which will help to recommend the exact types of a movie to pick for the similar attribute users.



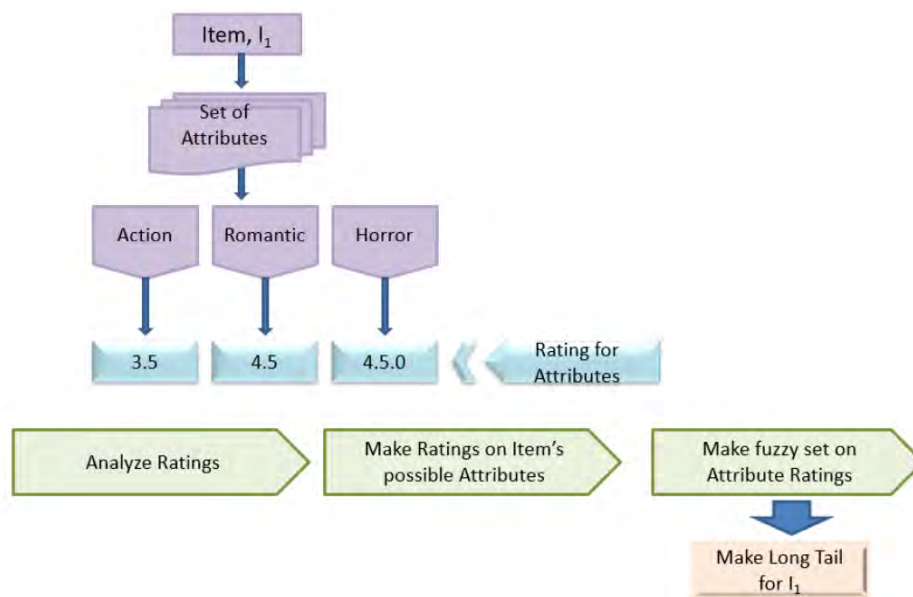


Figure 3.8: Item Profile Design Approach

Figure 3.8 is representing the model of item profile and in figure 3.9 is presenting the attribute's specification for crisp and fuzzy set. In general, one movie has some attributes and all those attributes are exist in that movie. But in general, other system is not quantify the attribute. Applying fuzzy set concept we take a initiate to quantify the attribute for a specific movie, that it contains.

---

#### Algorithm 3 Movie Rating Classification for Attribute

---

**Require:** Ratings of Movie for Attribute.

**Ensure:** Fuzzy set of Movie Ratings as degree of membership class.

**Step 1:** Get Attribute list of movie profile for each movie  $M_i$  with weight value  $R_a$  and compute  $RMS$  value of  $R_a$  as crisp input.

**Step 2:** Optimized *Rating for Attribute* of  $M_i$  and compute degree of membership class in *poor, low, medium, high*.

**Step 3:** Calculate the percentage value of  $M_i$ , from *Rating* value of *Attribute*  $R_a$   $RMS$  for each attribute.

**Step 4:** Sort the attribute's rating value in descending order and use for create long tail.

**Step 5:** Save the data in three output as degree of membership class,  $RMS$  value and percentage value.

---

#### Example of Movie Attribute List

Considering *Avengers* as Item as Movie and some Users to give feedback as they preferred to. Following the algorithm 3 step by step and classify the Movie rating is described bellow.

**Step 1:** Considering a movie *Avengers*, and its rating from different user  $U_t(4.015)$ ,  $U_b(3.64)$ ,  $U_z(3.75)$ .

$U_{t_{Avengers}} = (Action=3.5, Thriller=4.0, Sci-Fiction=4.0, Fantasy=4.5)$



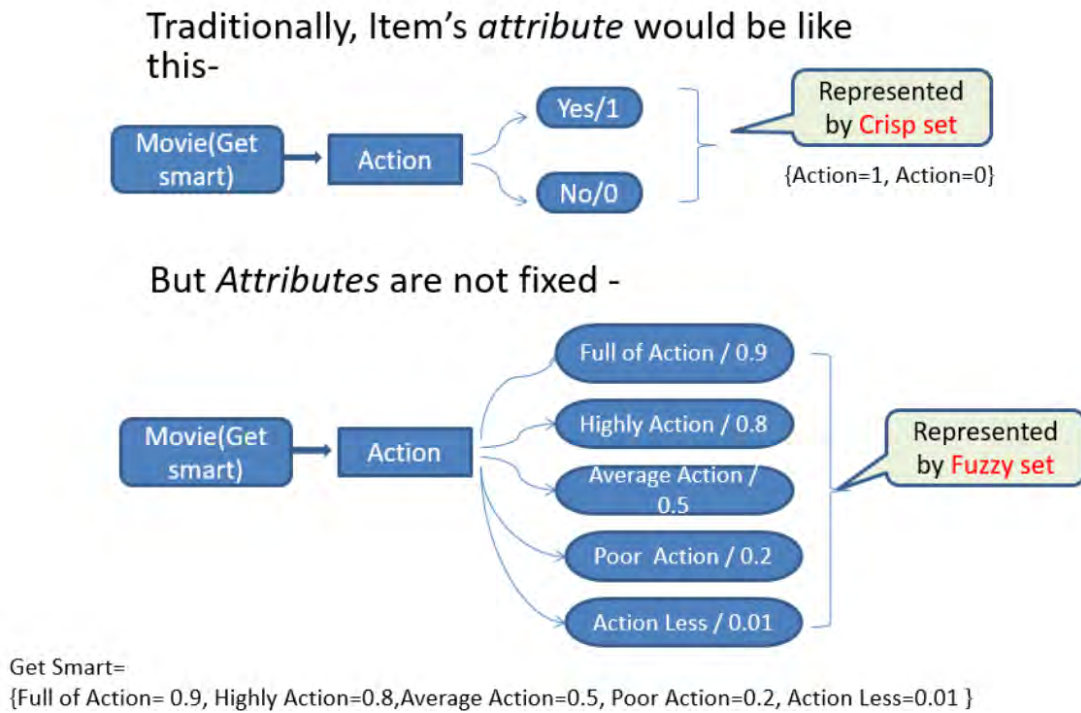


Figure 3.9: Item Attribute's fuzzy Specification

$$Ub_{Avengers} = (\text{Action}=3.0, \text{Animation}= 3.5, \text{Thriller}=4.0, \text{Fantasy}= 4.0)$$

$$Uz_{Avengers} = (\text{Sci-Fiction}= 4.0, \text{Mystery} = 3.5)$$

**Step 2:** Now the classified set of movie *Avengers* attribute's rating, calculate Root Mean Square value.

$$Avengers_{RMS} = (\text{Action}=3.25, \text{Thriller} =4.0, \text{Mystery}=3.5, \text{Animation}=3.5, \text{Sci-Fiction} =4.0 , \text{Fantasy}= 4.25)$$

**Step 3:** Convert the classified set to fuzzy set of *Avengers* using fuzzy set theory.

$$Avengers_{fuzzyset} = ( \text{Action}=0.325, \text{Thriller} =0.40, \text{Mystery}=0.35, \text{Animation}=0.35, \text{Sci-Fiction}=0.40 , \text{Fantasy}= 0.425)$$

**Step 4:** Final choice list of user *Avengers* is in percentage and membership class.

$Avengers_{choicelist} = \text{Fantasy} \mid 85 \text{ percent} \mid \text{High},$   
 $\text{Sci-Fiction} \mid 80 \text{ percent} \mid \text{High},$   
 $\text{Thriller} \mid 80 \text{ percent} \mid \text{High},$

Animation | 70 percent | Medium,  
 Mystery | 70 percent | Medium,  
 Action | 65 percent | Medium

This choice list of user will be used to measure the degree of attribute, match with user's Attribute List for recommendation.

### 3.4 Recommendation Methodology

Our proposed recommendation methodology is applying the fuzzy set theory for the attribute of user profile and item profile. According to algorithm 1 and 2, there will be a user choice list of the attributes with weight. From the user choice list, the weight of attribute will be compared with the weight of item as the movie's attribute list. For our recommendation system, there is a predefined thresh-hold value,  $T$  to 0.30. This value is at 0.30 which will refer that the minimum 50 percent to 60 percent preference will be recommended to the user-item matched attributes.

From the basic concept of fuzzy inference system, we applied *Rule-based fuzzy inference system*. Our recommendation methodology can be described in two approaches, first one is for new user-item profile and second one is active user-item profile.

**1. For new user** User choice list of attribute and weight will compute from the preliminary selected attribute during registration.

**2. Active user** The attribute choice list of user will be updated as adding attribute with weight from his/her feedback as rating. Minimum 20 ratings on different movie as it's attribute will be considered for adding attribute to choice list.

**Movie attribute** As considering movie, a new movie will get weighted attribute in the time of upload.

According to time, and ratings new attribute would be added to movie's attribute list.

From the user attribute set, the degree of membership class will be multiplied by the movie's degree of membership class. If the result value is greater than thresh-hold value, then the movie will be recommended to that user. And the result value indicate the minimum preference level of user to movie. This will also tell about the membership class which means how much this attribute belongs to this movie and user choice list. Now, the recommendation methodology for user and movie is given bellow.

**Step 1:** Take the single user's fuzzy classified set of attribute.

**Step 2:** Compute the shorted attribute list and find the movie list corresponding to maximum similar attribute.

**Step 3:** According to attribute, multiply the degree of membership value according to user and

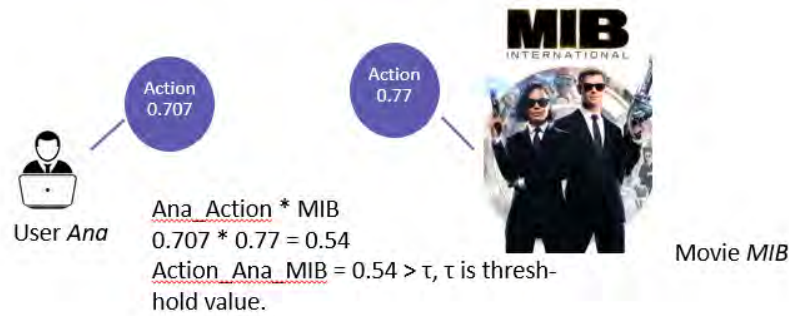


Figure 3.10: User choice's to movie with fuzzy classification

movie.

**Step 4:** If the computed value is greater or equal to the threshold value  $T$ , then the movie will be recommended to the user.

**Step 5:** The computed value will define the minimum preference value of choice by the user and indicate the predicted class of choice.

**Example** There is an example of movie recommendation for a single user given below.

If there is a new movie *MIB* with the attribute value of *Action* 0.707, and the user *Ua* has degree of attribute value is 0.77 for *Action* attribute, so how much this movie would be liked by user *Bob* or not, can be compute as,

$$MIB_{Action} \times Ua_{Action} = 0.707 \times 0.77 = 0.54, \text{ so } Ua_{Action-MIB} = 0.54.$$

The value of  $Ua_{Action-MIB}$  is greater than the predefined threshold value  $T$  which is 0.30, then the *MIB* movie will be recommended to user *Ua*, shown in figure 3.10.

Each movie will be classified using rating as their attribute consistency and then there will be a long tail for each attribute which will help to recommend the exact types of a movies to pick for the user who has the similar interest.

## 3.5 Data Collection

To apply our new classified recommendation methodology, we build a customized website with interface for taking input value of movie type, user, movie name, etc. From online registration section, new user can create profile with basic information and can add the preferred attribute from the attribute list. From this list, user will get suggestions from top movies of a matched attribute list. The admin user uploads movie with core attribute of the movie and then general users give feedback as rating on those attribute and the user can add more attribute anytime.

From this web site we collect attribute list, user preferred attribute list with rating, movie data with rating for our analysis. In general each movie has single rating for feedback, but in consideration of our classified method the rating will be added for each attribute. The name of our web site is 'Movie Recommendation' using *MySql* database for data management. We collect 100 data-set of user-item rating on different attribute. The next steps of the experiment are performed on this collected dataset. From the rating of users on movie for attributes, the user-item rating matrix is given on table as example.

User	Movie	A1(Action)	A2(Thriller)	A3(Drama)	A4(Romance)	A5(Horror)
$U_1$	$M_1$	5	5	2	2	0
$U_2$	$M_1$	4.5	4	1.5	2	1
$U_3$	$M_1$	4	5	2	1	0
$U_4$	$M_1$	5	4.5	1	1.5	0
$U_5$	$M_2$	2	2	4	4	4
$U_6$	$M_2$	1.5	1.5	3.5	4.5	4
$U_n$	$M_n$	...	...	...	...	...

Table 3.2: Table of User Item rating matrix

From the above user-item rating matrix table, the rating process is easy to understand and afterward, the rating of the individual attribute is counted for user and the item both for which rating class it should belong to.

$$X_{RMS} = \sqrt{\frac{\sum (X_1^2 + X_2^2 + X_3^2 + \dots + X_N^2)}{N}} \quad (3.1)$$

Then, *Standard Deviation* is computed to detect the rating accuracy.

$U_1M_1C_1, W_1(\text{Action}) = RMS$  of  $A_1$  value is computed from the equation,

$$U_1M_1C_2, W_1(\text{Action} - M_1)_{RMS} = 4.643$$

$$U_1M_1C_3, W_1(\text{Thriller} - M_1)_{RMS} = X$$

$$U_1M_1C_4, W_1(\text{Drama} - M_1)_{RMS} = Y$$

Where,  $U$  is user,  $M$  is for movie,  $C$  for choice in the list of users,  $W$  is the weight of rating.

The *RMS* value of each attribute corresponds to the user and item or movie gets input to the fuzzy inference system (*Rule – based Mamdani Model*) as a crisp set. Here, the *RMS* value is giving better average estimation. *Mamdani* and *Assilian* [17] models work with input data with the minimal or no tuning. Our proposed method with some modification provides better estimation. In Figure 2.2, fuzzy inference technique has been used with *Mamdani* model.

*Mamdani* model has good efficiency for getting a *crisp set* from *the fuzzy set* as output with shorted in descending order. This output has been used to build the *Long Tail* graph for each attribute with their item. We calculate the *similarity* using the fuzzy multiplication operator of

the user profile's 'proficiency of attribute' and 'movie attribute'. We then compare with the threshold value  $T$  which is predefined as 0.30. Figure 3.10 the movie recommendation for user.

By the time of using the model of user profile gets matured by adding more attributes and then it recommends the user the most desirable item or movie.

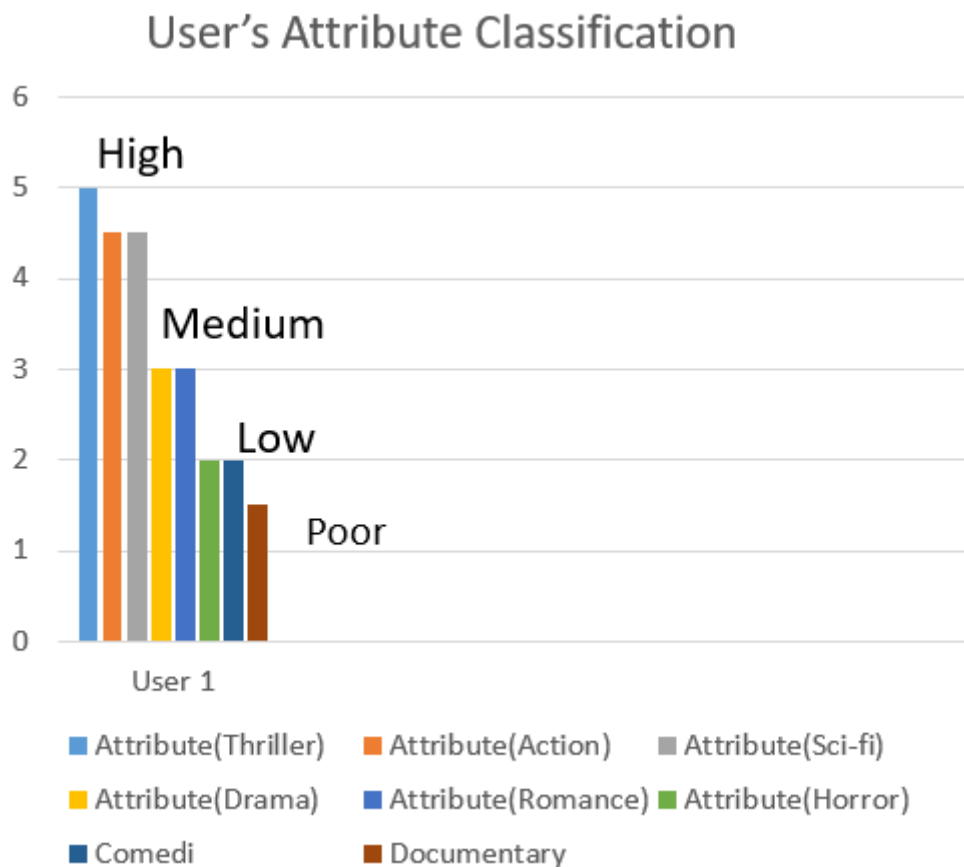


Figure 3.11: User choice class representation and the long tail.

Figure 3.11 represents the user's preferred choices. For a single user, each user's ratings will be counted a class and a sorted list of rating and the long tail is created.

Figure 3.12 represents the movie's attributes which have been in their specific class. The above example has been done for a single movie. However, each movie's ratings is computed in this way. The long tail is generated from sorted rating. From overall rating, the attribute's consistency is measured in numeric percentage which decides the movie category.

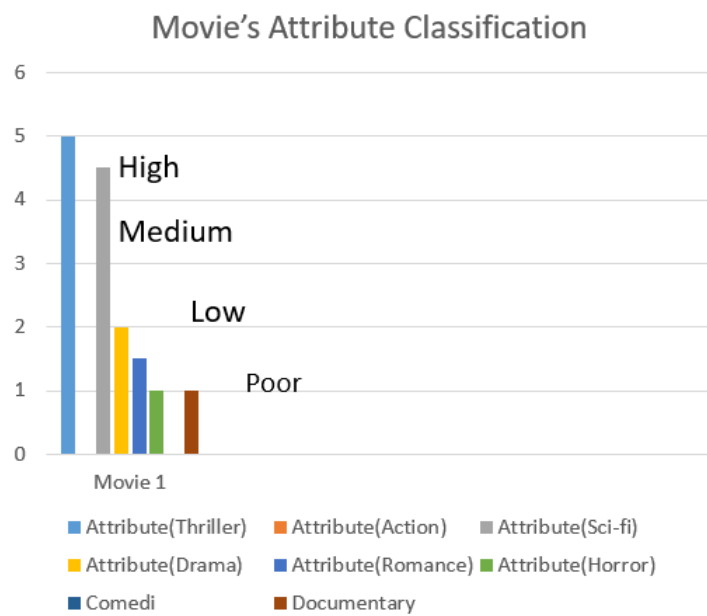


Figure 3.12: Movie attribute class representation and the long tail.

# Chapter 4

## Results and Analysis

In this chapter, we provide the experimental results and analysis. Here, user-item rating frequencies with attribute class labeling are reported. Furthermore, the comparison results with different classification algorithms on popular *Movielence* data-sets from the *Grouplence* organization [18] are also reported here. In *Movielence* data-set, there is only one class label for each movie, however, in our proposed method movie are labeled as multiple classes (according to attribute classification)

We report items' attribute classification based on the user choices. Different performance metrics are stated as follows. *Standard Deviation*, *Recall*, *Precision* etc. have been used to measure the performance of our proposed recommendation system.

In our experiment, we have considered one hundred users feedback on seventy movies. These movies are classified based on the attributes of the movies. To verify the rating by users, we consider the minimum number (20) from one user and minimum twenty ratings for one movie. We also consider the standard deviation value for user's rating valuation precisely. Furthermore, the rating from users for movie is used to recommend from their fuzzy set classification and the degree of membership.

For our data set, measuring the value of standard deviation is important because it measures the spread out of rating values from an average value. *Standard Deviation* is a number which is used to tell how the measurements for a group spreads out from the average (mean), or expected value. A low standard deviation means that most of the numbers are closed to the average. A high standard deviation means that the numbers are more spread out. *Standard Deviation* can be computed by taking the square root of the variance, which itself is the average of the squared differences of the mean [19]. In our system, we calculate the standard deviation for user's ratings on attributes. This is used for measure the rating authenticity of user which ensure that active

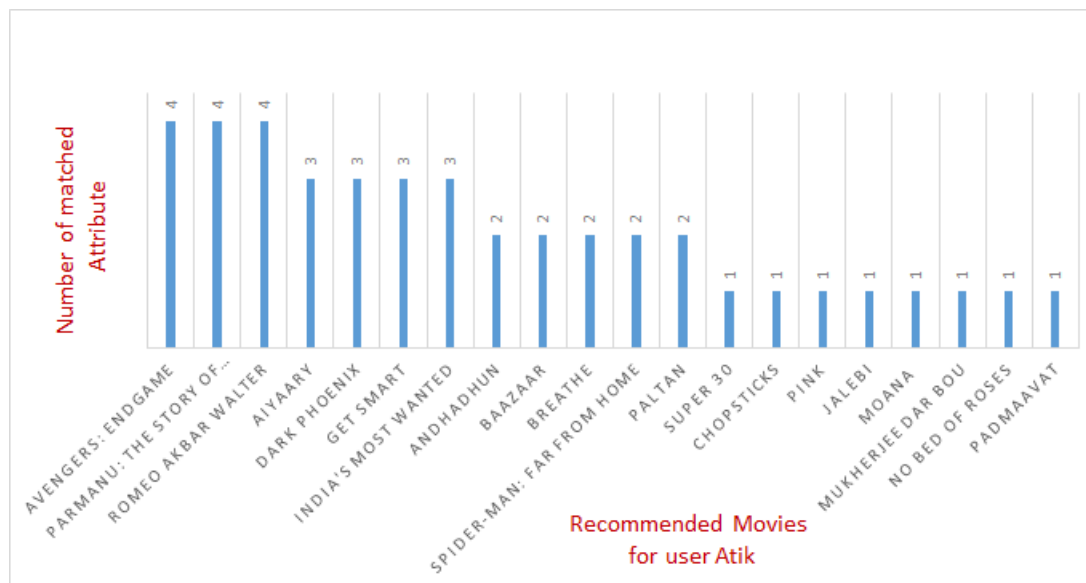


Figure 4.1: Recommended movie for single user.

users are put their feedback appropriately.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}.$$

Table 4.1: Standard deviation

Rating	Mean	Standard Deviation	Variance
Attribute ( $A_i$ )	4.9375	0.165	0.0272
Personalized User( $U_i - A_i$ )	4.3235	0.593	0.3516
Movie ( $M_i$ )	4.1765	.64	0.4096

Table 4.2: Standard Deviation for 'Movielence' data-set

Rating	Mean	Standard Deviation	Variance
User $_i$	4.2414	0.773	0.5975
Movie ( $M_i$ )	4.1552	0.778	0.6053

In Table 4.1, we take 35 rating of users for our new data-set and in Table 4.2, we take 35 rating of user from *Movielence* data-set. From these two tables we can see that, the standard deviation and variance are more in *Movielence* data-set.

From our web application we collect user's choice according to movies attribute and we find that users receive recommendation matching their choices or according to the given attributes. User can see the number of matched attribute. Figure 4.1 is a representation of recommended matched movie for a single user with his or her choice list of attributes.

Here we can see, this user get recommended first three movies with four matched attributes, then



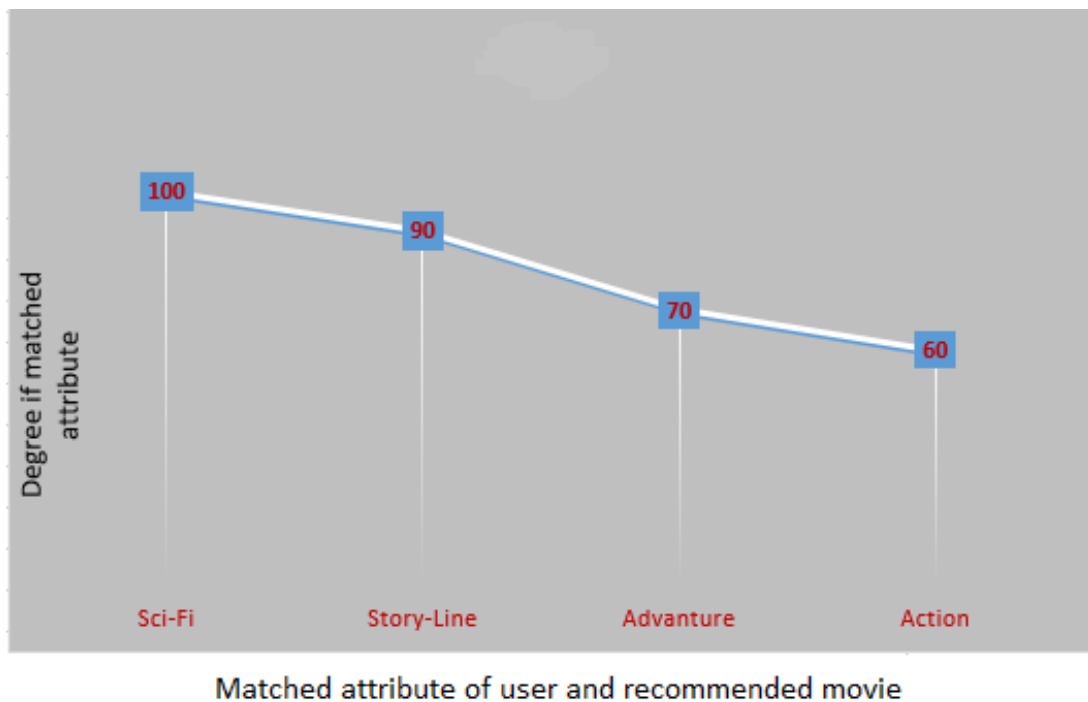


Figure 4.2: Single movie with attribute match for single user.

three and rest as the minimum matching. Our system is performing this result to make decision for users so that the movie will be the best suitable matching for him/her. Users can see only the matched and familiar movie, he/she can take one from this list. If user wants to see some new attributes for the movie, user can take one from the *tail of list*.

Figure 4.2 shows the detail of attribute matching quantity with users. For example,  $U_1$  to movie  $M_7$ , named as *Avengers:End Game*, where  $U_1$  is highly matched with attribute *Sci – Fiction*, then *StoryLine*, *Adventure*, and *Action* as quantifying into 100, 90, 70, and 60 percent. In this way, user can find the suitable movie from these detail information which is very simply represented.

#### 4.0.1 Impact of Attribute Quantification

From the analysis of some existing movies such as *Paltan*, which has two matched attribute with  $U_1$ , this user may not be willing to enjoy this movie. But this movie has total five attribute with degree of attribute attached in table 4.3.

This *Paltan* movie has rating 5.2 out of 10 in *IMDB* website and only 29 percent is the recommended probability as popular in *Rotten Tomato* site. Which means, when a viewer review this rating he/she unwilling have watched this movie. But this movie is full of war environment and military activities and also it is a true story of *Sikkim war* plot of 1962. If this movie is watched by those users who love to watch true story or historical movies, then this will get the

<i>Attribute Id</i>	<i>Attribute Name</i>	<i>Degree of Attribute</i>
1	Action	3
2	Adventure	4
8	Drama	4.5
17	War	5
25	History	5

Table 4.3: Table of degree of attribute for a single movie *Paltan*.

popularity as it deserves. Sometimes, some users who has the *Low* level of attribute choice in *History*, he/she might be interested to watch this movie from this attribute quantification.

There is also given a list of recommended movies to the users who love action type movie in compared with *Movielence* data-set and our new produced data-set. Where we can see the difference of number of recommended movies with attribute quantification.

Sometimes, with a good *IMDB* rating suppose above of six out of ten, movies are getting "flop" and a very few people know about that.

	IMDB	Rotten Tomato	New RecSys
Movie			
Paltan	5.2 (10) Action Drama History	29%	Action 3 Adventure 4 Drama 4.5 War 5 History 5 True Story 5
India's Most Wanted	4.4 (10) Action Thriller	29%	Action 3.5 Thriller 4 Story-Line 4 Performance 4
Romeo Akbar Walton	6.5 (10) Action Drama Thriller	17%	Action 4 Adventure 4 Thriller 4.5 Story-Line 5 Biography 4.5
Breathe	7.1 (10) Biography Drama Romance	68%	Biography 5 Adventure 4.5 Romance 4 Art-Film 4.5 True Story 5 Drama 3 Story-Line 5

Table 4.4: Attribute Quantification

Table 4.4 is a comparison scenario of our proposed recommendation approach and different popular online platforms for movie recommendation. We can see that our quantification of movie attribute is more specific than others and this will help users to pick the right item or movie. From the preliminary rating or review percentage of movie is completely discouraged any user to see this movie. But using our the attribute quantification of this movie, new viewers can be added which it deserves.

We upload fifteen flop movies from different movie list of Bollywood and Hollywood recently released in 2018 and 2019, to get the feedback from the users as they got recommended of those movies. As from our attribute quantification, these movies are presented to users and the users are put their feedback using rating. They also added attribute as they felt. Some movies got one or two additional attribute with the degree of attribute.

#### 4.0.2 Impact on the Tail

Now, the comparison of recommended movie to a user who loves watching good movies and also is a active user corresponding to online platform. Here, we have taken some same movie data-set form *Movielence*, *IMDB*, and some other movie sites and our collected data set. In *Movielence* and other sites have single rating on a movie, but in our collected set has weighted rating for different attributes. For example, we have taken a movie which has 5.2 out of 10 rating in *IMDB* and can not earn the budget limit, where budget was 14 crore rupi, earned less then 10 crore rupi. And this movie can not reach the people's crowd.

Figure 4.3 is recommended the movie list to user *Robin*, where the movie *Paltan* is in 30<sup>th</sup> position in top 40<sup>th</sup> movie according to single rating.

Figure 4.4 is recommended the movie list to user *Robin*, with his attribute choice list. And the movie is in 3<sup>rd</sup> position in top 40<sup>th</sup> movie list, with *Five* matched attribute from 8 attribute of his list.

From this above comparison, we can see that, this new recommendation systems' recommended movie is so near to user choice. The algorithm also counts the degree of attribute given by users and movie attribute for precise computation.

Our system also has a good impact on *niche market* strategy and positioning. Many niche movies will able to meet the budget with this recommendation to appropriate user.

Aspects of this new data-set, there is 25 user who has some off-pick attributes such as *History*, *War*, *TrueStory*, *Art – Film*, *Story – Line*, *Performance*, etc. with pick attributes such as *Action*, *Adventure*, *Thriller*, etc. After the recommendation of movie using this methodology, 80% Users are like these movie and add some new attribute with rating. So we can say that, this system will able to sustain the niche market.

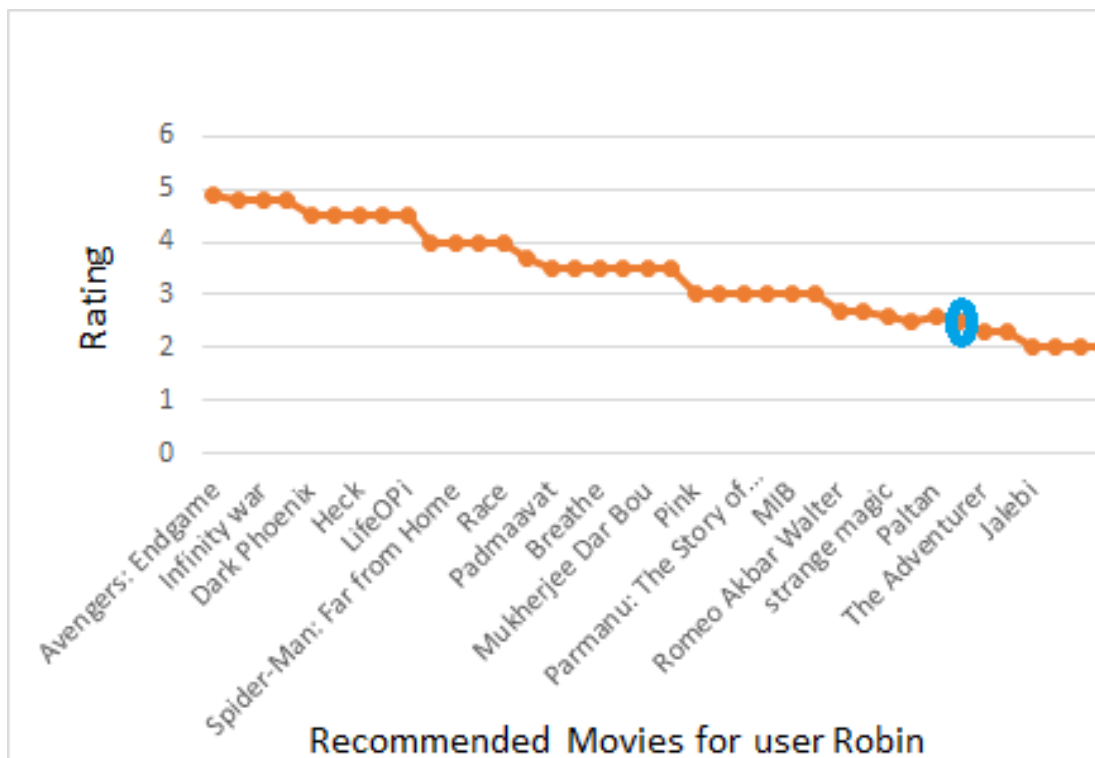


Figure 4.3: Recommended to User *Robin* followed by single rating.

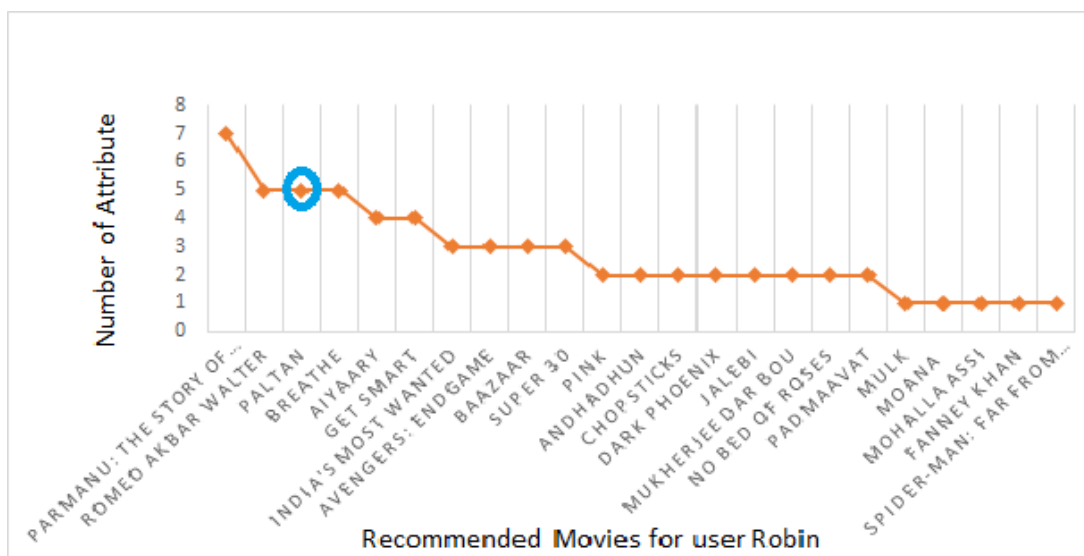


Figure 4.4: Recommended to User *Robin* followed by attribute rating.

### 4.0.3 User and Movie Class

Figure 4.5 and 4.6 represents the attribute wise user and item proficiency area, where the attribute part is more specific to represent about the profile of users and movies. We can see from Figure 4.5 and 4.6, is that, there is a pi-chart representation of  $User_1$  and  $Movie_1$  with the attribute existence. We find different colors for different attributes and we can see that one user has many choices in different measurement. Also one movie may have many attributes.

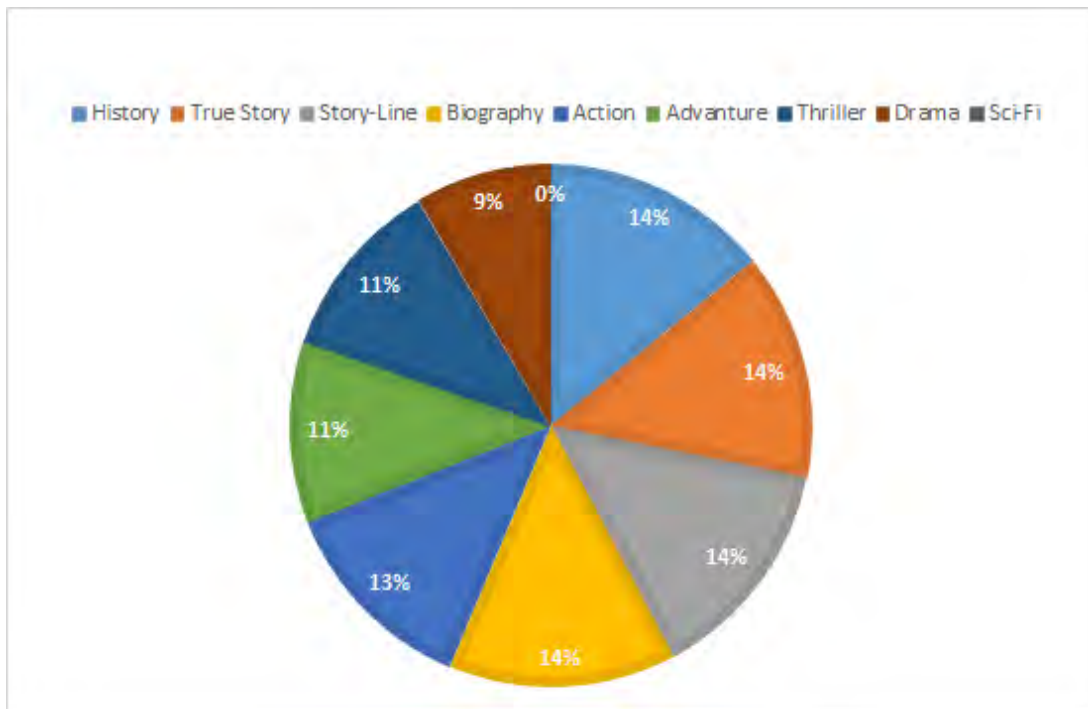


Figure 4.5: Attribute class of the user.

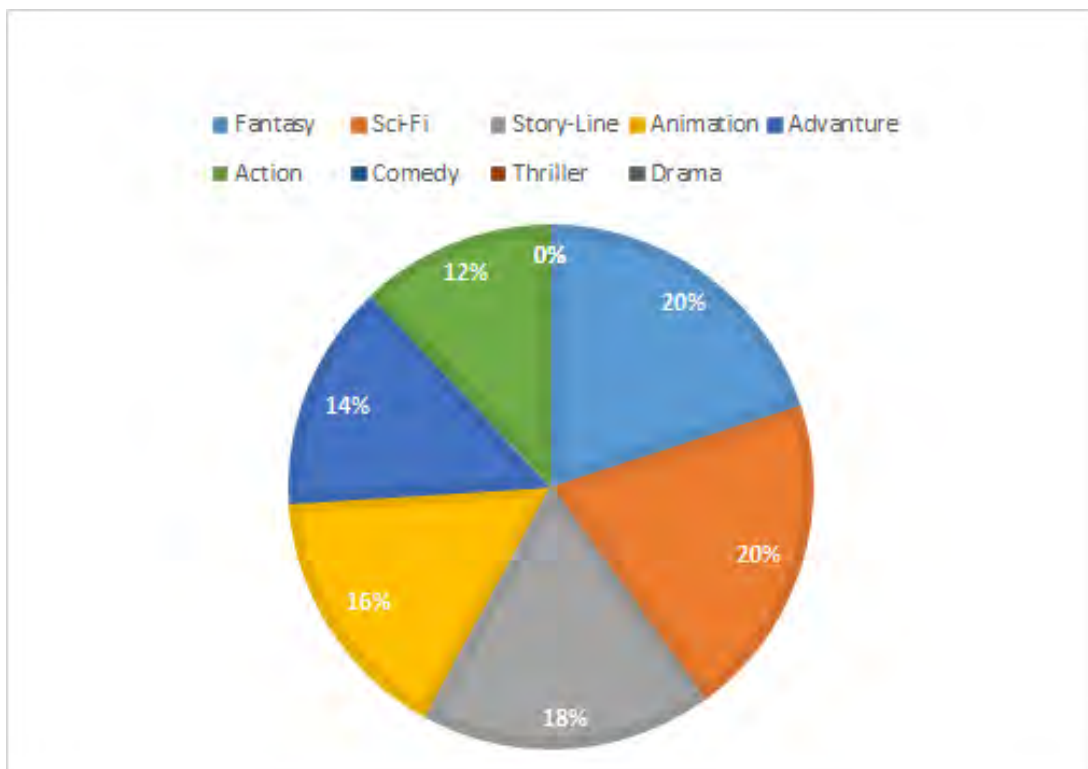


Figure 4.6: Attribute class of the user.

Here, we present some performance metrics which we used to measure the performance of the system.

We have used the data mining tool Rapidminer Stdio 9.3.1 on data-set [18] and have compared

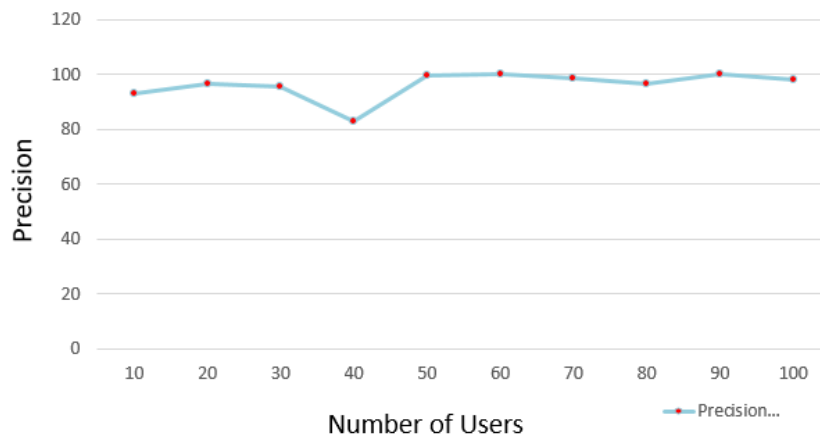


Figure 4.7: Precision value for proposed method

the proposed method to the performance for different algorithms. The value of Precision, Recall, and F-measure are also used to measure the performance.

We provide the classification results as follows.

The value of Precision, Recall, and F-measure are also summarized in the corresponding table.

Table 4.5: Table of Performance matrix of proposed method

User	Precision %	Recall %	F1 Measure %
10	93.175	80.64	86.455
20	96.815	90.9	94.21
30	95.723	76.92	85.296
40	83.134	81.10	82.10
50	99.648	83.91	91.104
60	100	87.11	93.111
70	98.595	90.9	94.591
80	96.82	71.42	82.2
90	100	80.66	89.294
100	97.861	90.19	93.869

Precision at  $N$  is the proportion of recommended movies in the top- $N$  set that are relevant. Here,  $N$  is set on 10 as the top-10 movies are recommended. In our experiment, precision at 10 in a top-10 recommendation problem is 90.90% for the first 10 users. This means that 90.90% of the recommendation this new system make are relevant to first 10 users choice.

In classification performance measurement the precision value is calculated based on four possible indicators of predicted rating, namely true positive (TP), false positive (FP), true negative (TN), and false negative (FN) [12].

$$\text{Precision} = TP / (TP + FP)$$

The graph showing the precision value in fig:4.7.

*Recall at  $N$*  is the proportion of relevant movies found in the top- $N$  recommendations. Here,  $N$  is

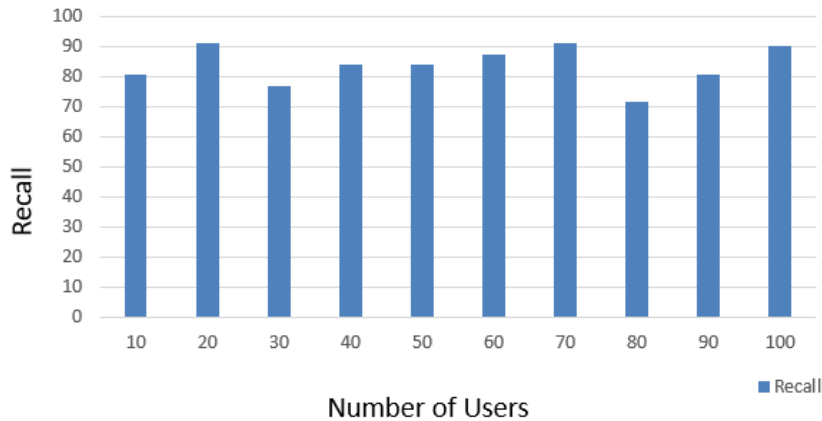


Figure 4.8: Recall value for the proposed method.

set on 10 as the top-10 movies are recommended. In our experiment, we computed recall at 10 and found it is 80.64% in our top-10 recommendation of the movie. This means that 80.64% of the total number of the relevant movies appear in the top- $N$  results, where  $N$  is set on 10.

In classification performance measurement the recall value is calculated based on four possible indicators of predicted rating, namely true positive (TP), false positive (FP), true negative (TN), and false negative (FN) [12].

$$Recall = TP / (TP + FN)$$

Figure 4.8, shows the recall value for proposed data-set.

*F1 Measure* is the weighted average of Precision and Recall. It is a statistical measure of the accuracy of a test or an individual. With an increase in the number of neighbors, the precision value decreases, while the recall value increases. Hence, the harmonic mean of both the precision and recall is accounted for evaluating the F1- measure. A higher F1 value signifies a high quality of prediction [12].

$$F1Measure = (2 \times precision \times recall) / (Precision + Recall)$$

In Figure 4.9, different f-measure values for proposed data-set are shown.

## 4.1 Impact on Attribute Coverage

A good recommendation system has a vast dependency on an attribute of user choices on the items. The basic goal is to cover the user choices attribute with the maximum matched items with those attributes. In this scenario, new methodology is working to obtain the maximum attribute coverage with high efficiency to satisfy the users.

For niche market establishment, *Segmentation*, *Target marketing*, and *Positioning* are the most important phases which are in combinedly known as *STP* model. The customer or user has four

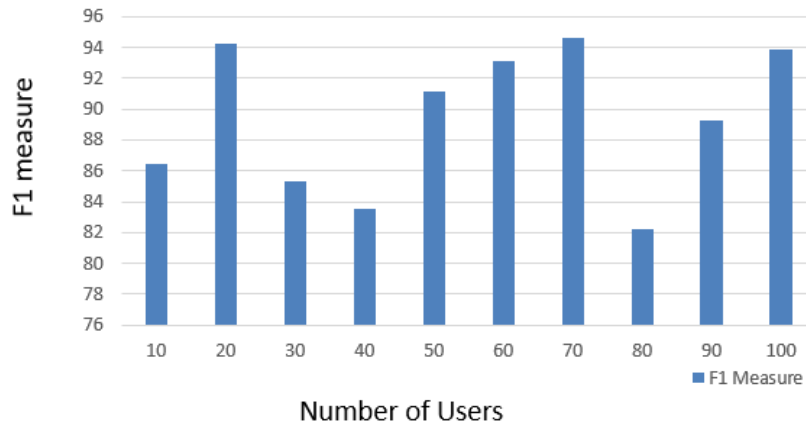


Figure 4.9: F-measure value for proposed method

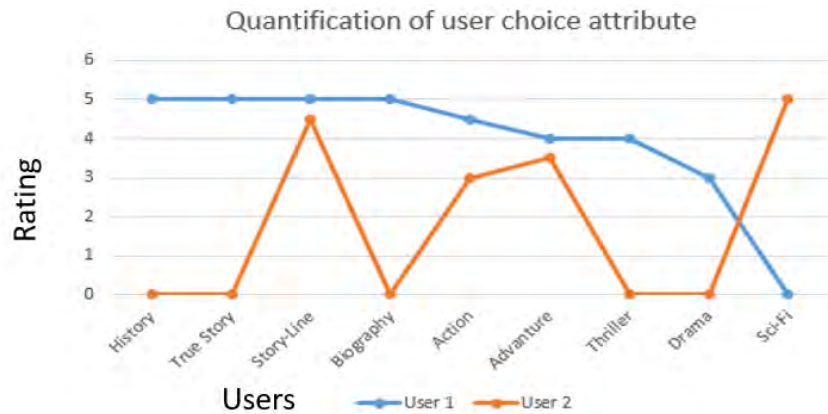


Figure 4.10: Attribute quantification of different users.

criteria of segmentation. These are *Demographic*, *Geographic*, *Behavioral*, and *Psycho-graphic*. Basically attribute coverage is graphical representation of the existence of attribute values in dataset. Our methodology focused on niche items as less popular movies, which has the potentiality to be popular as they deserve. We have applied segmentation on users considering their attribute preferences.

Figure 4.10 shows the attribute impact on different users. Here, we present two users  $User_1$  and  $User_2$  with their attribute choices. These two user has three common attribute with different weights. And they have also their individual attributes with weight. These users gets recommendation of movie with their similar attribute choices.

Figure 4.11 presents the attribute consistency of different movie. Here, we have considered two movies:  $Movie_2$  and  $Movie_7$ . The curves for these two movies are very dissimilar. Only two attributes are similar with different weights and they could be recommended to the same users having much common attribute choices.



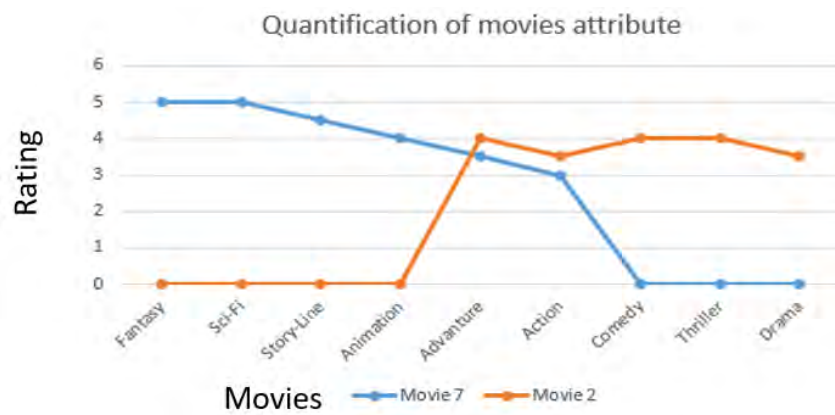


Figure 4.11: Attribute quantification of different movies.

This methodology has focused on niche market items with attributes along with the popular mainstream attributes too. The users who are not willing to go through trending movies such as full of action, thriller, etc., and they will not get good recommendations from some of mainstream popular movies.

From all above experimental results, we find that our proposed recommendation model using fuzzy set concept provide better recommendations of niche market items.

# Chapter 5

## Conclusions

In this thesis, we have proposed a recommendation system for the items of niche market based on fuzzy rating. We have considered movie recommendation system. We have developed interactive recommendation platform which provides the maximum satisfaction with weighted attribute matching of the items. We find that this system reduces the time in the long tail and helps in resolving *cold start* problem.

Our method can be applied to online book stores, video or content sharing sites etc. In our proposed model, long tail can be handled in such a way that niche items spend less time in tail position. Therefore, classified items can reach classified users quickly with the maximum business revenue generation.

# References

- [1] F. Isinkaye, Y. Folajimi, and B. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.
- [2] N. Vaidya and A. Khachane, “Recommender systems-the need of the ecommerce era,” in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 100–104, IEEE, 2017.
- [3] “Springer link.” <https://link.springer.com/referenceworkentry/>. Last Accessed: 2019-09-09.
- [4] R. Dunford, Q. Su, and E. Tamang, “The pareto principle,” 2014.
- [5] C. Anderson, “Why the future of business is selling less of more,” 2006.
- [6] Y. Liu, Q. Xiong, J. Sun, Y. Jiang, T. Silva, and H. Ling, “Topic-based hierarchical bayesian linear regression models for niche items recommendation,” *Journal of Information Science*, vol. 45, no. 1, pp. 92–104, 2019.
- [7] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, “Challenging the long tail recommendation,” *Proceedings of the VLDB Endowment*, vol. 5, no. 9, pp. 896–907, 2012.
- [8] X. Hu, C. Zhang, M. Wu, and Y. Zeng, “Research on long tail recommendation algorithm,” in *IOP Conference Series: Materials Science and Engineering*, vol. 261, p. 012019, IOP Publishing, 2017.
- [9] M. M. Chowdhury, F. Tanvir, M. S. Rahman, M. M. Rahman, M. Al-Sahariar, and R. M. Rahman, “Audio gadget recommendation by fuzzy logic,” in *Computer Science On-line Conference*, pp. 266–276, Springer, 2019.
- [10] M. El Mohadab, B. Bouikhalene, and S. Safi, “Predicting rank for scientific research papers using supervised learning,” *Applied Computing and Informatics*, vol. 15, no. 2, pp. 182–190, 2019.
- [11] A. N. Nikolakopoulos, V. Kalantzis, E. Gallopoulos, and J. D. Garofalakis, “Factored proximity models for top-n recommendations,” in *2017 IEEE international conference on big knowledge (ICBK)*, pp. 80–87, IEEE, 2017.

- [12] S. Bag, S. Kumar, A. Awasthi, and M. K. Tiwari, “A noise correction-based approach to support a recommender system in a highly sparse rating environment,” *Decision Support Systems*, vol. 118, pp. 46–57, 2019.
- [13] E. Christakopoulou and G. Karypis, “Local item-item models for top-n recommendation,” in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 67–74, ACM, 2016.
- [14] D. Wu, J. Lu, and G. Zhang, “A fuzzy tree matching-based personalized e-learning recommender system,” *IEEE transactions on fuzzy systems*, vol. 23, no. 6, pp. 2412–2426, 2015.
- [15] D. Zhang, C.-H. Hsu, M. Chen, Q. Chen, N. Xiong, and J. Lloret, “Cold-start recommendation using bi-clustering and fusion for large-scale social recommender systems,” *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 239–250, 2013.
- [16] C. Wang, “A study of membership functions on mamdani-type fuzzy inference system for industrial decision-making,” 2015.
- [17] H. Sandya, P. Hemanth Kumar, and S. K. R. Himanshi Bhudiraja, “Fuzzy rule based feature extraction and classification of time series signal,” *Int. J. Soft Comput. Eng.(IJSCE)*, vol. 3, no. 2, pp. 2231–2307, 2013.
- [18] “Movielens.” <https://grouplens.org/datasets/movielens/>. Last Accessed: 2019-09-09.
- [19] T. Chai and R. R. Draxler, “Root mean square error (rmse) or mean absolute error (mae)?—arguments against avoiding rmse in the literature,” *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

# Appendix A

## Codes

### A.1 SQL Query Code 1

```
1 /*
2 Navicat Premium Data Transfer
3
4 Source Server      : MySQL_LOCAL
5 Source Server Type : MySQL
6 Source Server Version : 100135
7 Source Host        : localhost:3306
8 Source Schema      : movies
9
10 Target Server Type : MySQL
11 Target Server Version : 100135
12 File Encoding      : 65001
13
14 Date: 21/09/2019 22:13:32
15 */
16
17 SET NAMES utf8mb4;
18 SET FOREIGN_KEY_CHECKS = 0;
19
20 -- -----
21 -- Table structure for countries
22 -- -----
23 DROP TABLE IF EXISTS `countries`;
24 CREATE TABLE `countries` (
25   `id` int(11) NOT NULL AUTO_INCREMENT,
26   `name` varchar(500) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NU
```

```
27 `created_at` timestamp(0) NULL DEFAULT NULL,
28 `updated_at` timestamp(0) NULL DEFAULT NULL,
29 `created_by` int(11) NULL DEFAULT NULL,
30 `updated_by` int(11) NULL DEFAULT NULL,
31 PRIMARY KEY (`id`) USING BTREE
32 ) ENGINE = InnoDB AUTO_INCREMENT = 4 CHARACTER SET = utf8 COLLATE = utf8_general
33
34 -- -----
35 -- Records of countries
36 -- -----
37 INSERT INTO `countries` VALUES (1, 'Bangladesh', NULL, NULL, NULL, NULL);
38 INSERT INTO `countries` VALUES (2, 'India', NULL, NULL, NULL, NULL);
39 INSERT INTO `countries` VALUES (3, 'Pakisthan', NULL, NULL, NULL, NULL);
40
41 -- -----
42 -- Table structure for genres
43 -- -----
44 DROP TABLE IF EXISTS `genres`;
45 CREATE TABLE `genres` (
46 `id` int(11) NOT NULL AUTO_INCREMENT,
47 `genre_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL,
48 `created_at` timestamp(0) NULL DEFAULT NULL,
49 `updated_at` timestamp(0) NULL DEFAULT NULL,
50 PRIMARY KEY (`id`) USING BTREE
51 ) ENGINE = InnoDB AUTO_INCREMENT = 21 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_
52
53 -- -----
54 -- Records of genres
55 -- -----
56 INSERT INTO `genres` VALUES (2, 'Action', '2019-09-21_14:28:18', NULL);
57 INSERT INTO `genres` VALUES (3, 'Adventure', '2019-09-21_14:28:18', NULL);
58 INSERT INTO `genres` VALUES (4, 'Animation', '2019-09-21_14:28:18', NULL);
59 INSERT INTO `genres` VALUES (5, 'Childrens', '2019-09-21_14:28:18', NULL);
60 INSERT INTO `genres` VALUES (6, 'Comedy', '2019-09-21_14:28:18', NULL);
61 INSERT INTO `genres` VALUES (7, 'Crime', '2019-09-21_14:28:18', NULL);
62 INSERT INTO `genres` VALUES (8, 'Documentary', '2019-09-21_14:28:18', NULL);
63 INSERT INTO `genres` VALUES (9, 'Drama', '2019-09-21_14:28:18', NULL);
64 INSERT INTO `genres` VALUES (10, 'Fantasy', '2019-09-21_14:28:18', NULL);
65 INSERT INTO `genres` VALUES (11, 'Film-Noir', '2019-09-21_14:28:18', NULL);
66 INSERT INTO `genres` VALUES (12, 'Horror', '2019-09-21_14:28:18', NULL);
67 INSERT INTO `genres` VALUES (13, 'Musical', '2019-09-21_14:28:18', NULL);
```

```
68 INSERT INTO `genres` VALUES (14, 'Mystery', '2019-09-21_14:28:18', NULL);
69 INSERT INTO `genres` VALUES (15, 'Romance', '2019-09-21_14:28:18', NULL);
70 INSERT INTO `genres` VALUES (16, 'Sci-Fi', '2019-09-21_14:28:18', NULL);
71 INSERT INTO `genres` VALUES (17, 'Thriller', '2019-09-21_14:28:18', NULL);
72 INSERT INTO `genres` VALUES (18, 'War', '2019-09-21_14:28:18', NULL);
73 INSERT INTO `genres` VALUES (19, 'Western', '2019-09-21_14:28:18', NULL);
74 INSERT INTO `genres` VALUES (20, 'Unknown', '2019-09-21_14:47:26', NULL);
75
76 -- -----
77 -- Table structure for menus
78 -- -----
79 DROP TABLE IF EXISTS `menus`;
80 CREATE TABLE `menus` (
81   `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
82   `name` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
83   `menus_description` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode
84   `menus_type` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NUL
85   `parent_menus_id` int(11) NOT NULL,
86   `modules_id` int(11) NOT NULL,
87   `icon_class` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
88   `menu_url` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT N
89   `sort_number` int(11) NOT NULL,
90   `created_by` int(11) NOT NULL DEFAULT 0,
91   `updated_by` int(11) NOT NULL DEFAULT 0,
92   `created_at` timestamp(0) NULL DEFAULT NULL,
93   `updated_at` timestamp(0) NULL DEFAULT NULL,
94   `is_active` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
95   PRIMARY KEY (`id`) USING BTREE
96 ) ENGINE = InnoDB AUTO_INCREMENT = 6 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_u
97
98 -- -----
99 -- Records of menus
100 -- -----
101 INSERT INTO `menus` VALUES (1, 'Add_New_Movie', NULL, 'Main', 0, 2, 'fa_fa-plus'
102 INSERT INTO `menus` VALUES (5, 'All_Movies', NULL, 'Main', 0, 1, 'fa_fa-list', '
103
104 -- -----
105 -- Table structure for migrations
106 -- -----
107 DROP TABLE IF EXISTS `migrations`;
108 CREATE TABLE `migrations` (
```

```
109  `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
110  `migration` varchar(191) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
111  `batch` int(11) NOT NULL,
112  PRIMARY KEY (`id`) USING BTREE
113 ) ENGINE = InnoDB AUTO_INCREMENT = 3 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci
114
115 -- -----
116 -- Records of migrations
117 -- -----
118 INSERT INTO `migrations` VALUES (1, '2014_10_12_000000_create_users_table', 1);
119 INSERT INTO `migrations` VALUES (2, '2014_10_12_100000_create_password_resets_table', 1);
120
121 -- -----
122 -- Table structure for modules
123 -- -----
124 DROP TABLE IF EXISTS `modules`;
125 CREATE TABLE `modules` (
126  `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
127  `name` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
128  `modules_icon` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
129  `description` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
130  `home_url` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
131  `created_by` int(11) NOT NULL DEFAULT 0,
132  `updated_by` int(11) NOT NULL DEFAULT 0,
133  `status` varchar(500) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
134  `created_at` timestamp(0) NULL DEFAULT NULL,
135  `updated_at` timestamp(0) NULL DEFAULT NULL,
136  PRIMARY KEY (`id`) USING BTREE
137 ) ENGINE = InnoDB AUTO_INCREMENT = 3 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci
138
139 -- -----
140 -- Records of modules
141 -- -----
142 INSERT INTO `modules` VALUES (1, 'Admin', 'fff', 'na', '/', 0, 0, 'Active', NULL, NULL);
143 INSERT INTO `modules` VALUES (2, 'User', '', '', '', 0, 0, 'Active', NULL, NULL);
144
145 -- -----
146 -- Table structure for movie_ratings
147 -- -----
148 DROP TABLE IF EXISTS `movie_ratings`;
149 CREATE TABLE `movie_ratings` (
```



```
150 `id` int(11) NOT NULL AUTO_INCREMENT,
151 `movie_id` int(11) NULL DEFAULT NULL,
152 `genre_id` int(11) NULL DEFAULT NULL,
153 `rating` float(126, 0) NULL DEFAULT NULL,
154 `created_at` timestamp(0) NULL DEFAULT NULL,
155 `updated_at` timestamp(0) NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP(0),
156 `created_by` int(11) NULL DEFAULT NULL,
157 PRIMARY KEY (`id`) USING BTREE
158 ) ENGINE = InnoDB AUTO_INCREMENT = 10 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_
159
160 -- -----
161 -- Records of movie_ratings
162 -- -----
163 INSERT INTO `movie_ratings` VALUES (6, 9, 6, 5, '2019-09-21_13:00:13', '2019-09-
164 INSERT INTO `movie_ratings` VALUES (7, 9, 4, 5, '2019-09-21_19:17:30', '2019-09-
165 INSERT INTO `movie_ratings` VALUES (8, 8, 1, 5, '2019-09-21_19:18:28', NULL, 1);
166 INSERT INTO `movie_ratings` VALUES (9, 9, 6, 4, '2019-09-21_19:43:33', NULL, 1);
167
168 -- -----
169 -- Table structure for movies
170 -- -----
171 DROP TABLE IF EXISTS `movies`;
172 CREATE TABLE `movies` (
173 `movie_id` int(11) NOT NULL AUTO_INCREMENT,
174 `title` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL DEF
175 `poster_path` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NU
176 `created_at` timestamp(6) NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP(6),
177 `updated_at` timestamp(6) NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP(6),
178 `created_by` int(11) NULL DEFAULT NULL,
179 `updated_by` int(11) NULL DEFAULT NULL,
180 `description` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NULL,
181 PRIMARY KEY (`movie_id`) USING BTREE
182 ) ENGINE = InnoDB AUTO_INCREMENT = 10 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_
183
184 -- -----
185 -- Records of movies
186 -- -----
187 INSERT INTO `movies` VALUES (9, 'Pulp_Fiction', 'images/1569070813.png', '2019-0
188
189 -- -----
190 -- Table structure for password_resets
```

```
191 -- -----
192 DROP TABLE IF EXISTS `password_resets`;
193 CREATE TABLE `password_resets` (
194   `email` varchar(191) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL
195   `token` varchar(191) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL
196   `created_at` timestamp(0) NULL DEFAULT NULL,
197   INDEX `password_resets_email_index`(`email`) USING BTREE
198 ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci ROW_FORMAT =
199
200 -- -----
201 -- Table structure for privilege_levels
202 -- -----
203 DROP TABLE IF EXISTS `privilege_levels`;
204 CREATE TABLE `privilege_levels` (
205   `users_id` int(11) NULL DEFAULT NULL,
206   `user_levels_id` int(11) NULL DEFAULT NULL,
207   `created_at` timestamp(0) NULL DEFAULT NULL,
208   `updated_at` timestamp(0) NULL DEFAULT NULL,
209   UNIQUE INDEX `privilege_levels_user_id_user_level_id_unique`(`users_id`, `user
210 ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci ROW_FORMAT =
211
212 -- -----
213 -- Records of privilege_levels
214 -- -----
215 INSERT INTO `privilege_levels` VALUES (1, 1, NULL, NULL);
216 INSERT INTO `privilege_levels` VALUES (2, 2, '2019-07-05_20:09:30', '2019-07-05_
217
218 -- -----
219 -- Table structure for privilege_menus
220 -- -----
221 DROP TABLE IF EXISTS `privilege_menus`;
222 CREATE TABLE `privilege_menus` (
223   `menus_id` int(10) UNSIGNED NOT NULL,
224   `user_levels_id` int(11) NULL DEFAULT NULL,
225   `users_id` int(11) NULL DEFAULT NULL,
226   `all` tinyint(1) NOT NULL DEFAULT 0,
227   `create` tinyint(1) NOT NULL DEFAULT 0,
228   `edit` tinyint(1) NOT NULL DEFAULT 0,
229   `del` tinyint(1) NOT NULL DEFAULT 0,
230   `created_at` timestamp(0) NULL DEFAULT NULL,
231   `updated_at` timestamp(0) NULL DEFAULT NULL,
```

```

232  UNIQUE INDEX `privilege_menus_menu_id_user_level_id_unique`(`menus_id`, `user_
233 ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci ROW_FORMA
234
235 -- -----
236 -- Records of privilege_menus
237 -- -----
238 INSERT INTO `privilege_menus` VALUES (1, 1, NULL, 0, 0, 0, 0, NULL, NULL);
239 INSERT INTO `privilege_menus` VALUES (5, 1, NULL, 0, 0, 0, 0, NULL, NULL);
240
241 -- -----
242 -- Table structure for sys_dropdowns
243 -- -----
244 DROP TABLE IF EXISTS `sys_dropdowns`;
245 CREATE TABLE `sys_dropdowns` (
246   `id` int(10) NOT NULL AUTO_INCREMENT,
247   `dropdown_slug` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL D
248   `dropdown_mode` enum('dropdown','dropdown_grid') CHARACTER SET utf8 COLLATE ut
249   `sys_search_panel_slug` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_c
250   `sqltext` text CHARACTER SET utf8 COLLATE utf8_general_ci NULL,
251   `value_field` varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFA
252   `option_field` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DE
253   `multiple` tinyint(1) NULL DEFAULT 0,
254   `search_columns` text CHARACTER SET utf8 COLLATE utf8_general_ci NULL,
255   `dropdown_name` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NU
256   `description` text CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
257   `created_by` int(10) NULL DEFAULT NULL,
258   `created_at` datetime(0) NOT NULL DEFAULT CURRENT_TIMESTAMP(0),
259   `updated_by` int(10) NULL DEFAULT NULL,
260   `updated_at` datetime(0) NULL DEFAULT NULL,
261   `status` enum('Active','Inactive') CHARACTER SET utf8 COLLATE utf8_general_ci
262   PRIMARY KEY (`id`) USING BTREE,
263   UNIQUE INDEX `dropdownslug`(`dropdown_slug`) USING BTREE
264 ) ENGINE = InnoDB AUTO_INCREMENT = 4 CHARACTER SET = utf8 COLLATE = utf8_general.
265
266 -- -----
267 -- Records of sys_dropdowns
268 -- -----
269 INSERT INTO `sys_dropdowns` VALUES (1, 'country', 'dropdown', NULL, 'SELECT_id,
270 INSERT INTO `sys_dropdowns` VALUES (2, 'city', 'dropdown', NULL, 'SELECT_id, _nam
271 INSERT INTO `sys_dropdowns` VALUES (3, 'users', 'dropdown', NULL, 'SELECT_users.
272

```

```
273 -- -----
274 -- Table structure for user_genre
275 -- -----
276 DROP TABLE IF EXISTS `user_genre`;
277 CREATE TABLE `user_genre` (
278   `id` int(11) NOT NULL AUTO_INCREMENT,
279   `user_id` int(11) NULL DEFAULT NULL,
280   `genre_id` int(11) NULL DEFAULT NULL,
281   `created_at` timestamp(0) NULL DEFAULT NULL,
282   `updated_at` timestamp(0) NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP(0),
283   PRIMARY KEY (`id`) USING BTREE
284 ) ENGINE = InnoDB AUTO_INCREMENT = 4 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_u
285
286 -- -----
287 -- Records of user_genre
288 -- -----
289 INSERT INTO `user_genre` VALUES (1, 3, 2, '2019-09-21_11:49:29', '2019-09-21_11:
290 INSERT INTO `user_genre` VALUES (2, 3, 4, '2019-09-21_11:49:29', '2019-09-21_11:
291 INSERT INTO `user_genre` VALUES (3, 3, 6, '2019-09-21_11:49:29', '2019-09-21_11:
292
293 -- -----
294 -- Table structure for users
295 -- -----
296 DROP TABLE IF EXISTS `users`;
297 CREATE TABLE `users` (
298   `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
299   `name` varchar(191) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
300   `email` varchar(191) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL
301   `email_verified_at` timestamp(0) NULL DEFAULT NULL,
302   `password` varchar(191) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT N
303   `remember_token` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
304   `created_at` timestamp(0) NULL DEFAULT NULL,
305   `updated_at` timestamp(0) NULL DEFAULT NULL,
306   PRIMARY KEY (`id`) USING BTREE,
307   UNIQUE INDEX `users_email_unique`(`email`) USING BTREE
308 ) ENGINE = InnoDB AUTO_INCREMENT = 4 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_u
309
310 -- -----
311 -- Records of users
312 -- -----
313 INSERT INTO `users` VALUES (1, 'Foysal_Ahmed', 'nibir2k12@gmail.com', NULL, '$2y
```

```

314 INSERT INTO `users` VALUES (2, 'Sazid', 'nibir2k12@live.com', NULL, '$2y$10$a3co
315 INSERT INTO `users` VALUES (3, 'quraishi_sazid', 'sazid.mehtaz@gmail.com', NULL
316
317 SET FOREIGN_KEY_CHECKS = 1;

```

## A.2 SQL Query Code 2

```

1 -- phpMyAdmin SQL Dump
2 -- version 4.8.5
3 -- https://www.phpmyadmin.net/
4 --
5 -- Host: localhost:3306
6 -- Generation Time: Sep 22, 2019 at 12:29 AM
7 -- Server version: 10.2.27-MariaDB
8 -- PHP Version: 7.2.7
9
10 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
11 SET AUTOCOMMIT = 0;
12 START TRANSACTION;
13 SET time_zone = "+00:00";
14
15
16 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
17 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
18 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
19 /*!40101 SET NAMES utf8mb4 */;
20
21 --
22 -- Database: `nusaim_movies`
23 --
24
25 -- -----
26
27 --
28 -- Table structure for table `countries`
29 --
30
31 CREATE TABLE `countries` (
32   `id` int(11) NOT NULL,
33   `name` varchar(500) DEFAULT NULL,
34   `created_at` timestamp NULL DEFAULT NULL,

```

```
35  `updated_at` timestamp NULL DEFAULT NULL,
36  `created_by` int(11) DEFAULT NULL,
37  `updated_by` int(11) DEFAULT NULL
38 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPACT;
39
40 --
41 -- Dumping data for table `countries`
42 --
43
44 INSERT INTO `countries` (`id`, `name`, `created_at`, `updated_at`, `created_by`,
45 (1, 'Bangladesh', NULL, NULL, NULL, NULL),
46 (2, 'India', NULL, NULL, NULL, NULL),
47 (3, 'Pakisthan', NULL, NULL, NULL, NULL);
48
49 -- -----
50
51 --
52 -- Table structure for table `genres`
53 --
54
55 CREATE TABLE `genres` (
56  `id` int(11) NOT NULL,
57  `genre_name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
58  `created_at` timestamp NULL DEFAULT NULL,
59  `updated_at` timestamp NULL DEFAULT NULL
60 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=CO
61
62 --
63 -- Dumping data for table `genres`
64 --
65
66 INSERT INTO `genres` (`id`, `genre_name`, `created_at`, `updated_at`) VALUES
67 (2, 'Action', '2019-09-21_18:28:18', NULL),
68 (3, 'Adventure', '2019-09-21_18:28:18', NULL),
69 (4, 'Animation', '2019-09-21_18:28:18', NULL),
70 (5, 'Childrens', '2019-09-21_18:28:18', NULL),
71 (6, 'Comedy', '2019-09-21_18:28:18', NULL),
72 (7, 'Crime', '2019-09-21_18:28:18', NULL),
73 (8, 'Documentary', '2019-09-21_18:28:18', NULL),
74 (9, 'Drama', '2019-09-21_18:28:18', NULL),
75 (10, 'Fantasy', '2019-09-21_18:28:18', NULL),
```



```
117 (6, 'Chart', NULL, 'Main', 0, 1, 'fa_fa-list', '/charts', 3, 0, 0, NULL, NULL, '
118
119 -- -----
120
121 --
122 -- Table structure for table `migrations`
123 --
124
125 CREATE TABLE `migrations` (
126   `id` int(10) UNSIGNED NOT NULL,
127   `migration` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
128   `batch` int(11) NOT NULL
129 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=CO
130
131 --
132 -- Dumping data for table `migrations`
133 --
134
135 INSERT INTO `migrations` (`id`, `migration`, `batch`) VALUES
136 (1, '2014_10_12_000000_create_users_table', 1),
137 (2, '2014_10_12_100000_create_password_resets_table', 1);
138
139 -- -----
140
141 --
142 -- Table structure for table `modules`
143 --
144
145 CREATE TABLE `modules` (
146   `id` int(10) UNSIGNED NOT NULL,
147   `name` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
148   `modules_icon` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
149   `description` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
150   `home_url` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
151   `created_by` int(11) NOT NULL DEFAULT 0,
152   `updated_by` int(11) NOT NULL DEFAULT 0,
153   `status` varchar(500) COLLATE utf8mb4_unicode_ci NOT NULL,
154   `created_at` timestamp NULL DEFAULT NULL,
155   `updated_at` timestamp NULL DEFAULT NULL
156 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=CO
157
```



```
158 --
159 -- Dumping data for table `modules`
160 --
161
162 INSERT INTO `modules` (`id`, `name`, `modules_icon`, `description`, `home_url`,
163 (1, 'Admin', 'fff', 'na', '//', 0, 0, 'Active', NULL, NULL),
164 (2, 'User', '', '', '', 0, 0, 'Active', NULL, NULL);
165
166 -- -----
167
168 --
169 -- Table structure for table `movies`
170 --
171
172 CREATE TABLE `movies` (
173   `movie_id` int(11) NOT NULL,
174   `title` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
175   `poster_path` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
176   `created_at` timestamp(6) NULL DEFAULT NULL ON UPDATE current_timestamp(6),
177   `updated_at` timestamp(6) NULL DEFAULT NULL ON UPDATE current_timestamp(6),
178   `created_by` int(11) DEFAULT NULL,
179   `updated_by` int(11) DEFAULT NULL,
180   `description` text COLLATE utf8mb4_unicode_ci DEFAULT NULL
181 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
182
183 --
184 -- Dumping data for table `movies`
185 --
186
187 INSERT INTO `movies` (`movie_id`, `title`, `poster_path`, `created_at`, `updated_at`,
188 (9, 'Pulp_Fiction', 'images/1569070813.png', '2019-09-21_23:02:24.099045', '2019-09-21_23:02:24.099045',
189
190 -- -----
191
192 --
193 -- Table structure for table `movie_ratings`
194 --
195
196 CREATE TABLE `movie_ratings` (
197   `id` int(11) NOT NULL,
198   `movie_id` int(11) DEFAULT NULL,
```

```
199  `genre_id` int(11) DEFAULT NULL,
200  `rating` float(126,0) DEFAULT NULL,
201  `created_at` timestamp NULL DEFAULT NULL,
202  `updated_at` timestamp NULL DEFAULT NULL ON UPDATE current_timestamp(),
203  `created_by` int(11) DEFAULT NULL
204 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
205
206 --
207 -- Dumping data for table 'movie_ratings'
208 --
209
210 INSERT INTO `movie_ratings` (`id`, `movie_id`, `genre_id`, `rating`, `created_at`
211 (6, 9, 6, 5, '2019-09-21_17:00:13', '2019-09-21_17:00:13', 1),
212 (7, 9, 4, 5, '2019-09-21_23:17:30', '2019-09-21_17:00:13', 1),
213 (8, 8, 1, 5, '2019-09-21_23:18:28', NULL, 1),
214 (9, 9, 6, 4, '2019-09-21_23:43:33', NULL, 1),
215 (77, 9, 6, 2, '2019-09-21_22:25:20', '2019-09-21_22:25:20', 3),
216 (78, 9, 4, 2, '2019-09-21_22:25:20', '2019-09-21_22:25:20', 3),
217 (79, 9, 2, 3, '2019-09-21_22:25:20', '2019-09-21_22:25:20', 3),
218 (80, 9, 11, 2, '2019-09-21_22:25:20', '2019-09-21_22:25:20', 3),
219 (81, 9, 15, 3, '2019-09-21_22:25:20', '2019-09-21_22:25:20', 3);
220
221 -- -----
222
223 --
224 -- Table structure for table 'password_resets'
225 --
226
227 CREATE TABLE `password_resets` (
228  `email` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
229  `token` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
230  `created_at` timestamp NULL DEFAULT NULL
231 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
232
233 -- -----
234
235 --
236 -- Table structure for table 'privilege_levels'
237 --
238
239 CREATE TABLE `privilege_levels` (
```

```
240 `users_id` int(11) DEFAULT NULL,
241 `user_levels_id` int(11) DEFAULT NULL,
242 `created_at` timestamp NULL DEFAULT NULL,
243 `updated_at` timestamp NULL DEFAULT NULL
244 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
245
246 --
247 -- Dumping data for table `privilege_levels`
248 --
249
250 INSERT INTO `privilege_levels` (`users_id`, `user_levels_id`, `created_at`, `updated_at`)
251 (1, 1, NULL, NULL),
252 (2, 2, '2019-07-06_00:09:30', '2019-07-06_00:09:30');
253
254 -- -----
255
256 --
257 -- Table structure for table `privilege_menus`
258 --
259
260 CREATE TABLE `privilege_menus` (
261 `menus_id` int(10) UNSIGNED NOT NULL,
262 `user_levels_id` int(11) DEFAULT NULL,
263 `users_id` int(11) DEFAULT NULL,
264 `all` tinyint(1) NOT NULL DEFAULT 0,
265 `create` tinyint(1) NOT NULL DEFAULT 0,
266 `edit` tinyint(1) NOT NULL DEFAULT 0,
267 `del` tinyint(1) NOT NULL DEFAULT 0,
268 `created_at` timestamp NULL DEFAULT NULL,
269 `updated_at` timestamp NULL DEFAULT NULL
270 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
271
272 --
273 -- Dumping data for table `privilege_menus`
274 --
275
276 INSERT INTO `privilege_menus` (`menus_id`, `user_levels_id`, `users_id`, `all`, `create`, `edit`, `del`, `created_at`, `updated_at`)
277 (1, 1, NULL, 0, 0, 0, 0, NULL, NULL),
278 (5, 1, NULL, 0, 0, 0, 0, NULL, NULL),
279 (6, 1, NULL, 0, 0, 0, 0, NULL, NULL);
280
```

```
281 -- -----
282
283 --
284 -- Table structure for table `sys_dropdowns`
285 --
286
287 CREATE TABLE `sys_dropdowns` (
288   `id` int(10) NOT NULL,
289   `dropdown_slug` varchar(100) DEFAULT NULL,
290   `dropdown_mode` enum('dropdown','dropdown_grid') DEFAULT 'dropdown',
291   `sys_search_panel_slug` varchar(100) DEFAULT NULL,
292   `sqltext` text DEFAULT NULL,
293   `value_field` varchar(50) DEFAULT NULL,
294   `option_field` varchar(100) DEFAULT NULL,
295   `multiple` tinyint(1) DEFAULT 0,
296   `search_columns` text DEFAULT NULL,
297   `dropdown_name` varchar(100) NOT NULL DEFAULT '0',
298   `description` text NOT NULL,
299   `created_by` int(10) DEFAULT NULL,
300   `created_at` datetime NOT NULL DEFAULT current_timestamp(),
301   `updated_by` int(10) DEFAULT NULL,
302   `updated_at` datetime DEFAULT NULL,
303   `status` enum('Active','Inactive') DEFAULT 'Active'
304 ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;
305
306 --
307 -- Dumping data for table `sys_dropdowns`
308 --
309
310 INSERT INTO `sys_dropdowns` (`id`, `dropdown_slug`, `dropdown_mode`, `sys_search
311 (1, 'country', 'dropdown', NULL, 'SELECT_id,_name_FROM_countries', 'id', 'name',
312 (2, 'city', 'dropdown', NULL, 'SELECT_id,_name_FROM_cities', 'id', 'name', 0, NU
313 (3, 'users', 'dropdown', NULL, 'SELECT_users.id,_users.name_FROM_users_LEFT_JOIN
314
315 -- -----
316
317 --
318 -- Table structure for table `users`
319 --
320
321 CREATE TABLE `users` (
```

```

322  `id` bigint(20) UNSIGNED NOT NULL,
323  `name` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
324  `email` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
325  `email_verified_at` timestamp NULL DEFAULT NULL,
326  `password` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
327  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
328  `created_at` timestamp NULL DEFAULT NULL,
329  `updated_at` timestamp NULL DEFAULT NULL
330 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
331
332 --
333 -- Dumping data for table `users`
334 --
335
336 INSERT INTO `users` (`id`, `name`, `email`, `email_verified_at`, `password`, `remember_token`,
337 (1, 'Foysal_Ahmed', 'nibir2k12@gmail.com', NULL, '$2y$10$Wj2D050cOGnwGg05gsksJe3', NULL),
338 (2, 'Sazid', 'nibir2k12@live.com', NULL, '$2y$10$a3conL2XHkYqSmfkQYvkh01iFy9yQsj', NULL),
339 (3, 'quraishi_sazid', 'sazid.mehtaz@gmail.com', NULL, '$2y$10$hj3aBEF/7fmq0SoUv', NULL)
340
341 -- -----
342
343 --
344 -- Table structure for table `user_genre`
345 --
346
347 CREATE TABLE `user_genre` (
348  `id` int(11) NOT NULL,
349  `user_id` int(11) DEFAULT NULL,
350  `genre_id` int(11) DEFAULT NULL,
351  `created_at` timestamp NULL DEFAULT NULL,
352  `updated_at` timestamp NULL DEFAULT NULL ON UPDATE current_timestamp()
353 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci ROW_FORMAT=COMPACT
354
355 --
356 -- Dumping data for table `user_genre`
357 --
358
359 INSERT INTO `user_genre` (`id`, `user_id`, `genre_id`, `created_at`, `updated_at`)
360 (1, 3, 2, '2019-09-21_15:49:29', '2019-09-21_15:49:29'),
361 (2, 3, 4, '2019-09-21_15:49:29', '2019-09-21_15:49:29'),
362 (3, 3, 6, '2019-09-21_15:49:29', '2019-09-21_15:49:29');

```

```
363
364 --
365 -- Indexes for dumped tables
366 --
367
368 --
369 -- Indexes for table 'countries'
370 --
371 ALTER TABLE `countries`
372   ADD PRIMARY KEY (`id`) USING BTREE;
373
374 --
375 -- Indexes for table 'genres'
376 --
377 ALTER TABLE `genres`
378   ADD PRIMARY KEY (`id`) USING BTREE;
379
380 --
381 -- Indexes for table 'menus'
382 --
383 ALTER TABLE `menus`
384   ADD PRIMARY KEY (`id`) USING BTREE;
385
386 --
387 -- Indexes for table 'migrations'
388 --
389 ALTER TABLE `migrations`
390   ADD PRIMARY KEY (`id`) USING BTREE;
391
392 --
393 -- Indexes for table 'modules'
394 --
395 ALTER TABLE `modules`
396   ADD PRIMARY KEY (`id`) USING BTREE;
397
398 --
399 -- Indexes for table 'movies'
400 --
401 ALTER TABLE `movies`
402   ADD PRIMARY KEY (`movie_id`) USING BTREE;
403
```

```
404 --
405 -- Indexes for table 'movie_ratings'
406 --
407 ALTER TABLE `movie_ratings`
408   ADD PRIMARY KEY (`id`) USING BTREE;
409
410 --
411 -- Indexes for table 'password_resets'
412 --
413 ALTER TABLE `password_resets`
414   ADD KEY `password_resets_email_index` (`email`) USING BTREE;
415
416 --
417 -- Indexes for table 'privilege_levels'
418 --
419 ALTER TABLE `privilege_levels`
420   ADD UNIQUE KEY `privilege_levels_user_id_user_level_id_unique` (`users_id`, `us
421
422 --
423 -- Indexes for table 'privilege_menus'
424 --
425 ALTER TABLE `privilege_menus`
426   ADD UNIQUE KEY `privilege_menus_menu_id_user_level_id_unique` (`menus_id`, `use
427
428 --
429 -- Indexes for table 'sys_dropdowns'
430 --
431 ALTER TABLE `sys_dropdowns`
432   ADD PRIMARY KEY (`id`) USING BTREE,
433   ADD UNIQUE KEY `dropdownslug` (`dropdown_slug`) USING BTREE;
434
435 --
436 -- Indexes for table 'users'
437 --
438 ALTER TABLE `users`
439   ADD PRIMARY KEY (`id`) USING BTREE,
440   ADD UNIQUE KEY `users_email_unique` (`email`) USING BTREE;
441
442 --
443 -- Indexes for table 'user_genre'
444 --
```

```
445 ALTER TABLE `user_genre`
446   ADD PRIMARY KEY (`id`) USING BTREE;
447
448 --
449 -- AUTO_INCREMENT for dumped tables
450 --
451
452 --
453 -- AUTO_INCREMENT for table `countries`
454 --
455 ALTER TABLE `countries`
456   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
457
458 --
459 -- AUTO_INCREMENT for table `genres`
460 --
461 ALTER TABLE `genres`
462   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;
463
464 --
465 -- AUTO_INCREMENT for table `menus`
466 --
467 ALTER TABLE `menus`
468   MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
469
470 --
471 -- AUTO_INCREMENT for table `migrations`
472 --
473 ALTER TABLE `migrations`
474   MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
475
476 --
477 -- AUTO_INCREMENT for table `modules`
478 --
479 ALTER TABLE `modules`
480   MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
481
482 --
483 -- AUTO_INCREMENT for table `movies`
484 --
485 ALTER TABLE `movies`
```



```
486  MODIFY `movie_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
487
488  --
489  -- AUTO_INCREMENT for table `movie_ratings`
490  --
491  ALTER TABLE `movie_ratings`
492  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=82;
493
494  --
495  -- AUTO_INCREMENT for table `sys_dropdowns`
496  --
497  ALTER TABLE `sys_dropdowns`
498  MODIFY `id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
499
500  --
501  -- AUTO_INCREMENT for table `users`
502  --
503  ALTER TABLE `users`
504  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
505
506  --
507  -- AUTO_INCREMENT for table `user_genre`
508  --
509  ALTER TABLE `user_genre`
510  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
511  COMMIT;
512
513  /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
514  /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
515  /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```