Recommender System using Collaborative Filtering and Demographic Features

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By

Shano Solanki

(Roll No. 851232008)

Under the supervision of:

Dr. Shalini Batra

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Recommender System using Collaborative Filtering and Demographic Features*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Shalini Batra* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:

(Shano Solanki)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Shalini Batra)

Assistant Professor,

CSED

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. S. Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

ACKNOWLEDGEMENT

No volume of words are sufficient to express gratitude towards my guide **Dr. Shalini Batra**, Computer Science & Engineering Department, Thapar University, Patiala, who has been very considerate and cooperative and guided whole-heartedly. She has been a continuous source of motivation, encouragement and kind support for preparing this thesis report. It's all because of my guide that I was able to explore such a vast and new topic for me in a limited time. She shared with me all the innovative ideas to do the research in a right direction and in an organized manner.

I am also thankful to **Dr. S. S. Bhatia**, Dean of Academic Affairs, **Dr. Deepak Garg**, Head of Computer Science & Engineering Department and **Mr. Ashutosh Mishra**, P.G. Coordinator, for providing all the facilities and harmonious environment for learning.

I would also like to thank my colleagues for extending all kind of help and cooperation during thesis.

Most importantly, I would like to thank my parents, my family members and the almighty for showing me the way out of darkness and for giving me strength and intelligence to carry out this research work. This work would not have been possible without their support and patience and for being with me throughout my journey of exploration.

Shano Solanki

851232008

ABSTRACT

Recommender systems use variety of data mining techniques and algorithms to identify appropriate preferences of items for users in a system out of available millions of choices. Recommender systems are classified into Collaborative filtering (CF), Content-Based filtering (CBF) and Knowledge-Based filtering (KBF) and Hybrid filtering (HF) systems. In present scenario recommender systems are facing many challenges like data sparsity, cold start problem, scalability, synonymy, shilling attacks, gray sheep and black sheep problems etc. These problems and challenges consequently affect the performance of recommender systems to a great extent. Among these cold start problem is one of the challenges which comes into scene when either a new user enters into a system or a new product arrives in catalogue. Both situations lead to difficulty in predicting user preference in the absence of availability of sufficient user rating history. The research work in this thesis is based upon exploiting user demographic characteristics for finding similarity between new user and already existing users in the system. The efficiency of recommender systems can be improved by proposed Hybrid recommender system which calculates recommendations for new user on the basis of his / her demographic attributes like age, gender, occupation similarity of existing users in the system. The proposed approach results in generation of more relevant and accurate recommendations as compared to traditional methods of finding recommendations. The work is based upon problem domain related to online movie recommendation using MovieLens dataset.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Recommender System

Recommender systems are software applications or web portal that generates personalized preferences using information filtering techniques and algorithms with a goal to support decision making of the users. Recommender systems are extensively used in various application domains i.e. online books ordering, online shopping, online hotel bookings, audio and video recommendations and so on. The concept of recommendation is not new but in use from many years, the difference is due to more number of users asking for recommendations among thousands to millions of choices. It has become a tedious job to recommend someone appropriately without filtering the data for relevant choices. It depends upon several factors like users rating given to collection of items based upon their satisfaction level, their likes and dislikes, age, gender, occupation, region or locality, community *etc*. Some of the popular websites that are using recommendation engine to filter choices are listed below

Amazon, the popular e-commerce site, uses content-based recommendation. When you select an item to purchase, Amazon recommends other items other users purchased based on that original item (as a matrix of item-to-likelihood-of-next-item purchase) [1].

Hulu, a streaming-video website, uses a recommendation engine to identify content that might be of interest to users.
Netflix, the video rental and streaming service, is a famous example.

Other sites that incorporate recommendation engines include Facebook, Twitter, Google, MySpace, Last.fm, Del.icio.us, Pandora, Goodreads, and your favourite online news site. Considering the present scenario, one can say that use of a recommendation engine is becoming a standard element of a modern web presence [2].

1.2 Recommendation Techniques [3]

Recommender systems help to match users with items and for this various recommender systems have been designed according to availability of exploitable data, implicit and explicit user feedback, domain characteristics *etc*. Recommender Systems are classified according to approach/paradigm used for predicting preferences. Mainly these are classified into four types listed below:

- Collaborative filtering (CF)
- Content-Based Filtering (CBF)
- Knowledge-Based Filtering (KBF)
- Hybrid Filtering (HF)

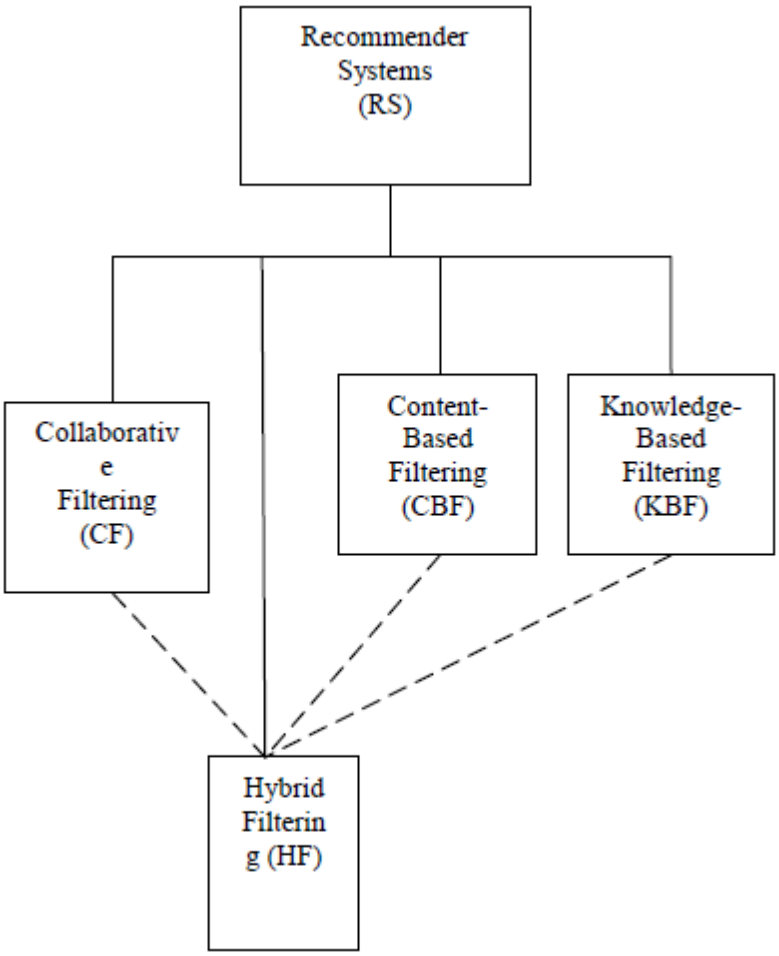

Figure 1: Classification of Recommender Systems

[3]

The simplest approach is to exploit user information if available directly to generate recommendations for user and this is referred to as personalized recommendations as shown below:

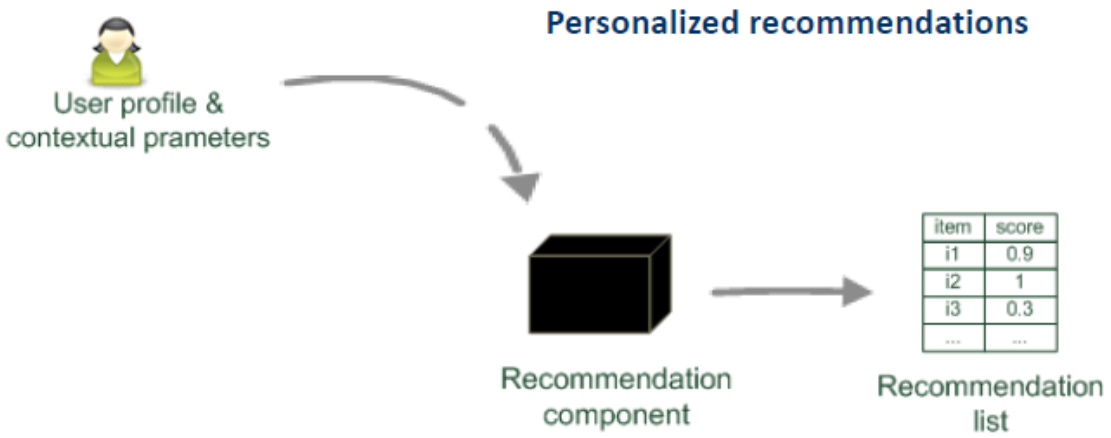

Figure 2: Personalized Recommender Systems [4]

1.2.1 Collaborative filtering based Recommender systems

Collaborative-Filtering systems focus on the relationship between users and items. [1]

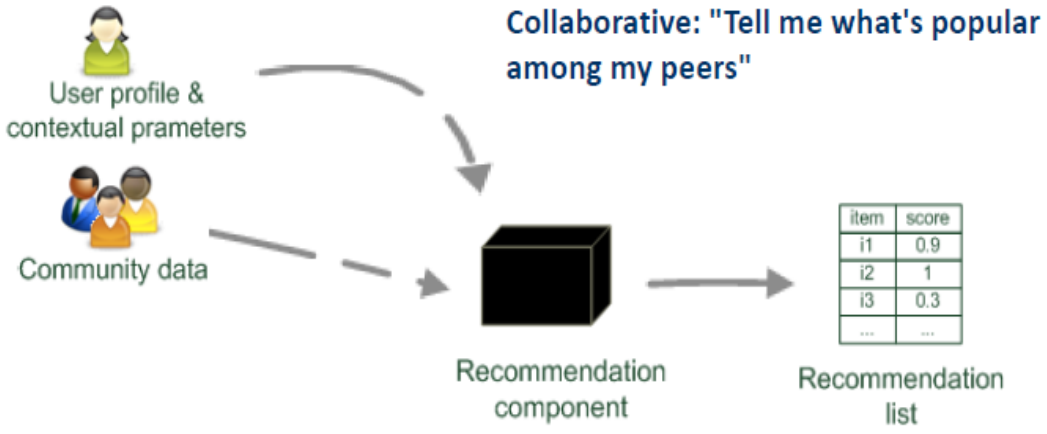

Figure 3: Collaborative filtering algorithm based Recommender Systems [4]

This method finds user-similarities using m x n rating matrix containing ratings data given by various users corresponding to same set of items in the online store. Here each entry in R user-item data matrix i.e. $R_{u,j}$ represents the rating given by user u to item j [3]. Collaborative filtering techniques are based upon either probabilistic models or non-probabilistic models. Nearest-neighbor algorithms are CF non-probabilistic algorithms. It comes in two different forms i.e. User-based nearest neighbor and item-based nearest neighbor collaborative filtering algorithms.

Table 1: User Rating Data Matrix R

	$Item_1$	$Item_2$	$Item_{\dots}$	$Item_i$	$Item_{\dots}$	$Item_n$
$User_1$	$R_{1,1}$	$R_{1,2}$	$R_{1,\dots}$	$R_{1,i}$	$R_{1,\dots}$	$R_{1,n}$
$User_2$	$R_{2,1}$	$R_{2,2}$	$R_{2,\dots}$	$R_{2,i}$	$R_{2,\dots}$	$R_{2,n}$
$User_{\dots}$	$R_{\dots,1}$	$R_{\dots,2}$	$R_{\dots,\dots}$	$R_{\dots,i}$	$R_{\dots,\dots}$	$R_{\dots,n}$
$User_u$	$R_{u,1}$	$R_{u,2}$	$R_{u,\dots}$	$R_{u,i}$	$R_{u,\dots}$	$R_{u,n}$
$User_{\dots}$	$R_{\dots,1}$	$R_{\dots,2}$	$R_{\dots,\dots}$	$R_{\dots,i}$	$R_{\dots,\dots}$	$R_{\dots,n}$
$User_m$	$R_{m,1}$	$R_{m,2}$	$R_{m,\dots}$	$R_{m,i}$	$R_{m,\dots}$	$R_{m,n}$

Further collaborative filtering is classified into two categories:

- Memory-based collaborative filtering techniques
- Model-based collaborative filtering techniques

1.2.1.1 Memory-based collaborative filtering techniques use the entire or a sample of the user-item database to generate predictions. Every user belongs to a group of users with similar interests. The approach is based upon finding neighbors of an active user or new user in order to predict his/her preferences on a new item. Here, we use

the nearest-neighbor based Collaborative filtering algorithms to predict the preferences or Top-N recommendations for the active user. Different kind of aggregate analysis can be performed on similar users' data to generate relevant recommendations, in certain priority order, if required.

Similarity computed can be item-based or user-based and there are several methods to calculate the similarity or distance or weight between users or items such as Minkowski distance, Pearson correlation and cosine-similarity metrics.

Minkowski distance: For two data objects, $X = (x1, x2, \ldots, xn)$ and $Y = (y1, y2, \ldots, yn)$, the popular Minkowski distance is defined as [5]

$$d(X, Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q},$$

----------(1)

Where n is the dimension number of the object and xi, yi are the values of the i^{th} dimension of object X and Y respectively, and q is a positive integer. When $q = 1$, d is *Manhattan* distance; when $q = 2$, d is *Euclidian* distance.

Correlation- Based Similarity: In this case, Pearson correlation measures the extent to which two variables linearly relate with each other. For user-based similarity, the Pearson correlation between two users say u and v is expressed as

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}},$$

---------- (2)

Where i Є I summations are over the items that both the users u and v have rated. Here $\bar{r}_u$ and $\bar{r}_v$ are the average rating of the co-rated items of the u^{th} and v^{th} users.

Similarly, for the item-based algorithm, denote the set of users u Є U who rated both items i and j

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}},$$

----------(3)

Where $r_{u,i}$ is the rating of user u on item i, $\bar{r}_i$ is the average rating of the i^{th} item by those users. Other correlation-based similarities include: *constrained Pearson correlation*, a variation of *Pearson correlation* that uses midpoint instead of mean rate; *Spearman rank correlation*, similar to *Pearson correlation*, except that the ratings are ranks; and *Kendall's τ correlation*, similar to the *Spearman rank correlation*, but instead of using ranks themselves, only the relative ranks are used to calculate the correlation. [3]

Vector Cosine-Based Similarity: *Vector cosine similarity* between items *i* and *j* is given by

$$w_{i,j} = \cos\left(\vec{i}, \vec{j}\right) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| * \|\vec{j}\|},$$

----------(4)

Where "•" denotes the dot-product of the two vectors. To get the desired similarity computation, for *n* items, an $n \times n$ similarity matrix is computed. For example, if the vector $\vec{A} = \{x1, y1\}$, vector $\vec{B} = \{x2, y2\}$, the vector cosine similarity between $\vec{A}$ and $\vec{B}$ is

$$w_{A,B} = \cos\left(\vec{A}, \vec{B}\right) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}.$$

-------(5)

So, the conclusion is, a choice is to be made among all similarity measures. The point to remember at this step is:

i) if the data is subject to grade-inflation i.e.(different users may be using different scales) then use pearson correlation coefficient [6].

ii) if data is dense i.e. (if almost all attributes have non-zero values) and the magnitude of the attribute value is important, use distance measures such as Euclidean or Manhattan.

iii) if the data is sparse consider using cosine-simialrity. [6]

1.2.1.2 Model-Based Collaborative Filtering Techniques:

Model-Based Collaborative filtering approach allows the system to learn to recognize complex patterns based on the training data, and then make intelligent predictions for the collaborative filtering tasks for test data or real-world data, based on the learned models. Sometimes complete dataset is taken as training data and sometimes the dataset is spilt in a particular ratio to train the model and for testing purpose. Bayesian models, clustering models are examples of model-based CF. Usually; classification algorithms can be used as *CF* models if the user ratings are *categorical*, and regression models and *SVD* methods and be used for *numerical ratings* [5].

1.2.2 Content-Based Filtering

This method finds the preferences of the current user about new item using rating history of current user related to previously used items. Similarity of items is determined by measuring the similarity in their properties. So, in this type of filtering method there is no dependency on rating records of other users in order to generate preferences for current user. Content-Based systems focus on properties of items. For example, if the user has purchased a book on amazon.com which uses recommender system then the user starts getting additional preferences for buying books from online book store which includes same or similar keywords information for books [3].

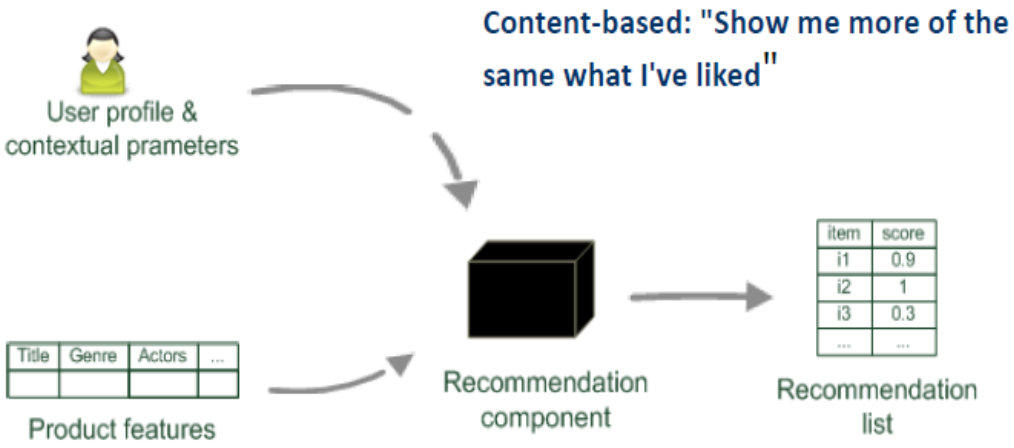

Figure 4: Content-based Recommender Systems [4]

1.2.3 Knowledge- Based Filtering

Knowledge based recommender systems makes use of knowledge structure to make inference about the user needs and preferences. Such kind of recommender systems have knowledge about what kind of items are liked by a user, so, a relationship can be established between user needs and relevant recommendation for that user [3].

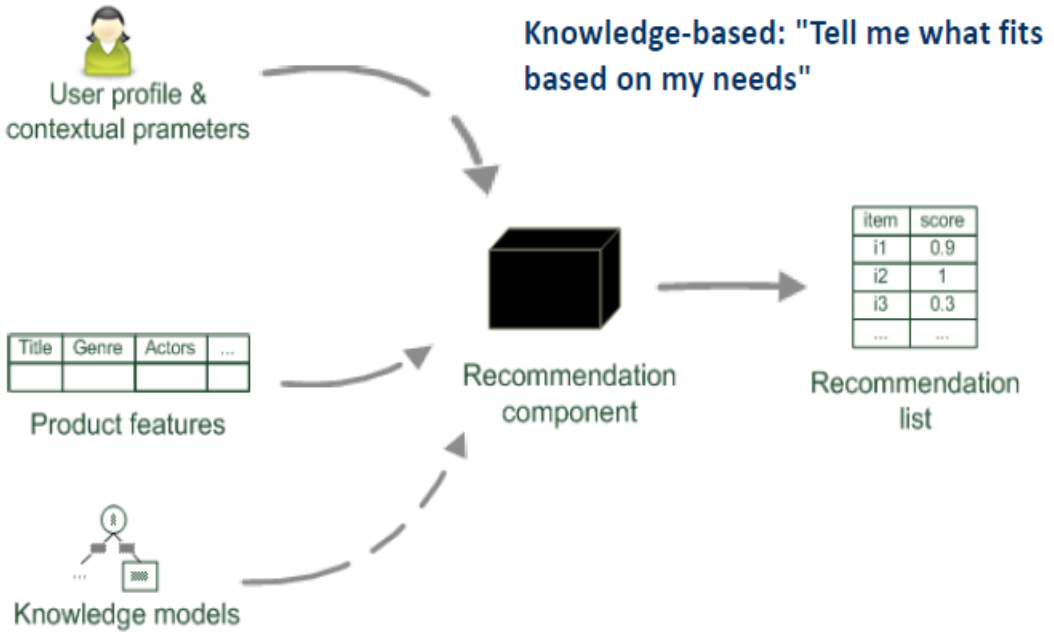

Figure 5: Knowledge-based Recommender Systems [4]

1.2.4 Hybrid Recommender Systems

No single recommender system approach is found to be efficient enough to generate relevant and accurate recommendation preferences, so, a hybrid recommender systems came into existence to overcome the limitations of traditional recommendation approaches mentioned above. These systems are based upon combining collaborative filtering approach with content-based approach or collaborative filtering with demographic characteristics based recommendation approach.

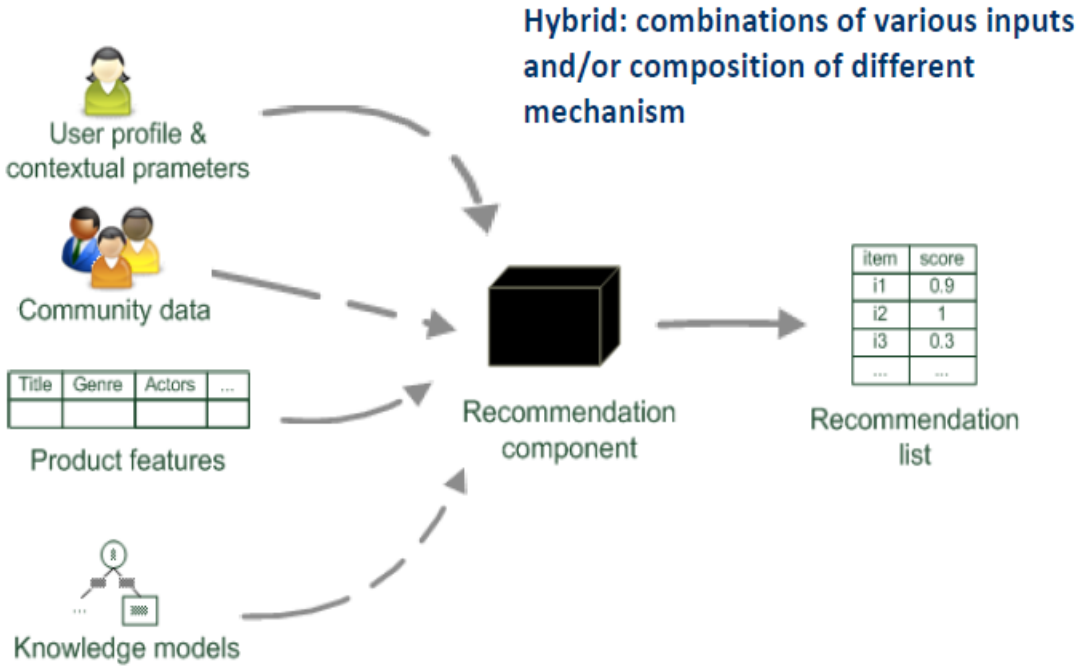

Figure 6: Hybrid Recommender Systems [4]

1.3 Recommendation Techniques Challenges

Recommender systems have been used extensively for generating recommendations using various recommendation techniques in multifarious application domains and this has brought into notice many challenges. Research areas are emphasizing on solving the issues mentioned below:

1.3.1 Data Sparsity

This problem arises if the user-item matrix containing ratings details is extremely sparse and this situation further leads to inefficient recommender systems which are based upon **nearest-neighbor algorithms** for calculating user similarity. This problem is further classified as **reduced coverage problem and neighbor transitivity problem** [3]**.**

- Reduced coverage means the recommender system is not able to produce recommendations including more number of items due to data sparsity as many of entries corresponding to items rating by various users is empty or not available.
- Neighbor transitivity problem arises when the users have not rated the similar set of items and it becomes difficult to find similar users for predicting preferences.

1.3.2 Scalability

This problem refers to a situation when numbers of items and users giving ratings those items increased tremendously and it becomes difficult for a recommender system to handle such a big data due to computational complexity and constrained resources. So, it goes beyond the limit of acceptability of recommender systems [3].

1.3.3 Synonymy

When some similar items have different names then recommender system fails to recognize that similarity and recommends them as if they are different items. This leads to problem of recommending similar items refers to as synonymy problem.

1.3.4 Gray sheep and black sheep problem:

These two are very prominent problems of recommender systems. The gray sheep problem arises because the user's choice does not match with any other user or group of users in agreement or disagreement consistently and on the other hand black sheep problem refers to a situation when the user's choice correlates with very few users or no users at all. In such cases recommender system proves to be inefficient or fruitless in generating preferences [3].

1.3.5 Shilling attacks

These attacks are categorized into **push attacks and nuke attacks.** When the competitor vender makes use of unfair ways and means to show more rating of their own items as compared to other venders products then it is known as push attacks. On the other hand if they try to reduce the rating of their rivals or competitors then it is called as nuke attacks.

1.3.6 Cold-start problem

When a recommender system is unable to generate or predict ratings due to initial lack of ratings then it is referred to as cold-start problem. This kind of situation happens when either a new user arrives into a system having no rating records available with recommender system or when a new item enters into system and till now no one has given rating to that item. So, it becomes difficult for a recommender system to generate choices for a new user and hence the objective of recommender systems is not achieved [7].

Besides all these problems discussed above, recommender systems suffers from many more challenges like generating preferences in cross-domains, context-aware recommendations, constrained based recommendations and many more [8].

1.3.7 Other challenges

In addition to above challenges of recommender system there arises issue of maintaining privacy of users information as one has to use it for collaborative filtering based recommendation generation and most of the time users are not willing to give their personal information for such tasks [5].

Another challenge is to filter the data using appropriate instances of data after removing noise. This process involves instances selection techniques based upon various criteria, for example, in MovieLens database only those instances are selected for experimentation in which a user has rated at least one movie. Similarly, other selection techniques can be applied for selection of instances for further testing.

1.4 Evaluation Metrics of Recommender Systems

The different type of CF based recommender systems use different metrics for evaluating their performance and hence provides information regarding the quality of recommender systems. Evaluation metrics plays a very important role in machine learning task. There are different metrics for the tasks of classification, regression, ranking, clustering, etc. Classification, regression, and ranking are examples of supervised learning, which constitutes the majority of machine learning applications. Diversity in recommendation preferences is essential for usefulness and user satisfaction of recommender system [9].

Classification is about predicting class labels given input data. In *binary classification*, there are two possible output classes. In *multi-class classification*, there are more than two possible classes.

Metrics used in recommender systems can be categorized into:

- Predictive accuracy metrics, such as Mean Absolute Error (MAE). It measures to what extent a system can predict ratings of users.
- Classification accuracy metrics measure how well a system is able to classify items correctly such as Precision, Recall, F1-measure and ROC Sensitivity

- Coverage metrics measure the percentage of items for which the system can make recommendations.
- Confidence metrics measure how certain the system is of the accuracy of the recommendations [10].
- Rank accuracy metrics such as Pearson's product-moment correlation, Kendall's Tau, Mean Average Precision (MAP), half-life utility and normalized-distance based performance metric(NDPM)

1.4.1 Mean Absolute Error (MAE) and Normalized Mean Absolute Error (NMAE) :

Mean Absolute Error (MAE) computes the average of the absolute difference between the predictions and true ratings [5]

$$MAE = \frac{\sum_{\{i,j\}} \left| p_{i,j} - r_{i,j} \right|}{n},$$

----------(6)

Here n is the total no. of ratings over all users , $p_{i,j}$ is the predicted rating for user i on item j and $r_{i,j}$ is the actual rating. **The lower the MAE, the better the prediction.**

Normalized Mean Absolute Error (NMAE) normalizes MAE to express errors as percentages of full scale

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}},$$

----------(7)

Here, r_{max} and r_{min} are the upper and lower bounds of the ratings and different recommender systems makes use of different numerical ratings as in the case of MovieLens database ratings for movies are chosen from 1-5 scale. The lowest rating is 1 and 5 is considered as highest rating [10]. These ratings are a primary source of predicting preferences, if available in dataset.

1.4.2 Root Mean Squared Error (RMSE)

It is one of the most popular performance measurements metric for recommender systems. It is expressed as

$$RMSE = \sqrt{\frac{1}{n}\sum_{\{i,j\}}\left(p_{i,j} - r_{i,j}\right)^2},$$

----------(8)

RMSE amplifies the contributions of the absolute errors between the predictions and true values [5].

1.4.3 Accuracy

Accuracy simply measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions (the number of test data points) [9].

Accuracy = no. of correct predictions / no. of total data points ----------(9)

A variation of accuracy is the *average per-class accuracy*--the average of the accuracy for each class. Accuracy is an example of a *micro-average*, and average per-class accuracy is a *macro-average*.

1.4.4 Precision-Recall-F Score

Precision and Recall metrics are usually used together. Precision means out of the items that the ranker/classifier predicted to be relevant, how many are truly relevant? On the other hand, recall measures out of all the items that are truly relevant, how many are found by the ranker/classifier?

Precision = no. of correct answers / no. of total items returned by ranker ----------(10)

Recall = no. of correct answers / no. of total relevant items ----------(11)

F-score is the harmonic mean between Precision and Recall. If Precision is high then Recall is low or vice-versa. F-measure is also referred to as the F1 measure in recommender system evaluation terminology [2]. It is defined as:

$$F\text{-}measure = \frac{2\,Precision\,Recall}{Precision + Recall} = \frac{2}{1/Precision + 1/Recall}$$

----------(12)

1.4 Structure of the thesis

The remaining thesis is structured based upon chapters as shown below:

Chapter 2: Literature Review. This chapter introduces the related work that has been done in the field of recommendation techniques, their challenges and limitations in generating relevant recommendations.

.**Chapter 3:** Problem Statement. In this chapter the gaps in the field of recommender system approaches and objectives of proposed research work has been described.

Chapter 4: Proposed Methodology and Technologies used. This chapter describes the proposed methodology to solve the cold-start problem of recommender systems using example of MovieLens dataset and brief detail of tools used.

Chapter 5: Introduction to Dataset and Experiment Results. This chapter contains the description of the results achieved against experiments conducted on MovieLens dataset and observations of the experimental analysis of proposed system.

Chapter 6: Conclusion and Future Scope. The whole work presented in thesis is summarized in this chapter and it also contains the scope for future research work in same or different problem domain.

Chapter 2

Literature Review

2.1 Evolution of Recommender Systems

People have been relying on recommender systems for different reasons in their day to day life. The concept of recommender system is not new as it has been into practice for a long time though offline since there are so many professions and businesses, and the success of which is totally dependent upon their recommender system. For example, the medical and legal professions are two such examples where recommender system is quite significant as the success of a doctor or a lawyer depends a lot upon the recommendations of their patients/clients.

The concept of online recommender system has gained popularity due to online availability of goods and services and the role of recommender system becomes more important when one kind of product or services are being offered by so many online service providers. Therefore recommender system has proved a boon for the society which helps them choose the best among the available choices. The recommender system uses several approaches for providing best results in the form of recommendations for an individual who is looking for preferences from an online recommender system. In the present times the following approaches are being used namely, Collaborative filtering based, Content-Based filtering, knowledge-based filtering and hybrid recommender systems which makes use of combination of traditional approaches.

2.2 Overview of Collaborative Filtering techniques

Recommender Systems are called as CF based recommender systems as it makes recommendations based on other people choices, likes/dislikes or ratings for the items they had used earlier. Presently many collaborative filtering based recommender systems software are freely available for research as described in [2] are listed below:

- Apache Mahout: Machine learning library which includes collaborative filtering (Java Based)

- Crab : Components to create recommender systems (Python based)
- LensKit : Collaborative filtering algorithms from from GroupLens
- easyrec: Recommender for web pages (Java Based)
- Vogoo PHP LIB: Collaborative filtering engine for personalizing web sites (PHP Based)

As explained by several representative techniques are used under collaborative filtering categories and each having their advantages and shortcomings as shown below in the table [5]:

Table 2: Overview of Collaborative Filtering Techniques

CF categories	Representative techniques	Main advantages	Main shortcomings
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations	*easy implementation *new data can be added easily and incrementally *need not consider the content of the items being recommended *scale well with co-rated items	*are dependent on human ratings *performance decrease when data are sparse *cannot recommend for new users and items *have limited scalability for large datasets
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, *SVD*, *PCA*	*better address the sparsity, scalability and other problems *improve prediction performance *give an intuitive rationale for recommendations	*expensive model-building *have trade-off between prediction performance and scalability *lose useful information for dimensionality reduction techniques
Hybrid recommenders	*content-based CF recommender, for example, *Fab* *content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis	*overcome limitations of CF and content-based or other recommenders *improve prediction performance *overcome CF problems such as sparsity and gray sheep	*have increased complexity and expense for implementation *need external information that usually not available

So, Collaborative filtering algorithms are divided into memory-based and model-based CF algorithms. Memory-based CF use whole or large sample of the dataset to create recommendations and hence results in scalability problem. User-based CF is the most

commonly used algorithm in this category. On the other hand, Model-based algorithms use the user database to learn a more compact model for example clusters of users with similar choices are found from dataset and later on recommendations are generated on the basis of information of users in a particular cluster [2].

Now a day's hybrid recommender systems are gaining popularity which helps in providing a better solution as compared to single recommendation approach.

2.3 Cold-start problem solution using various approaches

Cold-start problem of recommender system has two variations i.e. new user and new item. When the new user has few or no rating records available for generating recommendations is called as new-user cold-start problem. Similarly, if a new item which has just arrived in the system and it has not been rated by any user is called as new-item cold-start problem [11].

In [13] five different classification strategies are used, and then the returned results of the underlying classifiers are fused using Optimistic exponential type of ordered weighted averaging (OWA) operator. First approach utilizes user rating history for similarity computation. Second approach focuses on items attribute based similarity computation, which are already purchased by the user. Third approach makes use of demographic information contained in user profiles and previous ratings. Fourth approach is purely based upon the user demographic information. Finally, fifth approach applies both i.e. items attribute information of previous purchases and rating history of each customer. If final output of the recommender system in the form of predicted label of item x for the user y is 1, x will recommend to y, otherwise it will not recommend.

OWA operator (introduced by Yager) of dimension n is a mapping such as [11]:

$F: R^n \rightarrow R$ and is given by

$$OWA(a_1, a_2, \ldots\ldots, a_n) = \sum_{i=1}^{n} w_i b_i$$

Where b_i is the i^{th} largest element among a_i's. The weights are non-negative (wi >= 0) and their sum equals to one $(\sum_{i=1}^{n} w_i = 1)$.

The aggregation done by an OWA operator always is between the maximum and the minimum. In a degree of maxness (initially called orness) was introduced as follow:

$$Maxness\ (w_1, w_2, \dots\dots, w_n\ ) = \frac{\sum_{i=1}^{n}(n-i)w_i}{n-1}$$ ----------(13)

A simple class of OWA operators as exponential class was introduced to generate the OWA weights satisfying a given degree of maxness. The optimistic and pessimistic exponential OWA operators were correspondingly introduced as follows:

Optimistic:

$$w_i = \alpha \times (1-\alpha)^{i-1}, \forall\ i \neq n;\ w_n = (1-\alpha)^{n-1}$$ ----------(14)

Pessimistic:

$$w_1 = \alpha^{n-1};\ w_i = (1-\alpha) \times \alpha^{n-i}, i \neq 1$$ ----------(15)

Where parameter α belongs to the unit interval [0 1] and is related to orness value regarding the n. The proposed hybrid approach has shown improvement in the prediction accuracy of the existing recommender systems in the cold-start (new user) condition. This approach can further be improved by providing dynamicity to α which is used as a static parameter in this research study. Further the same approach can be tested for solving other problems in recommender system.

In [7] demographic information based recommender system is proposed which utilizes users demographic data stored in their profiles like Age, Gender, Occupation etc. to find similarity. It assumes that users with similar demographic attribute(s) will rate items similarly. Hence, a neighborhood is composed of users with similar demographic characteristics and that has been further utilized for generating recommendations. The proposed framework aims at evaluating how demographic attributes affects the user ratings and assists the recommender system in improving recommendation quality for new user i.e. deals effectively with cold-start problem. The whole recommendation process is divided into three stages: data input, similarity calculation and recommendation generation. It is quite essential for a recommender system to know appropriately the criteria for checking user or item similarity.

The framework consists of four modules i.e. data source, attribute analysis, splitting dataset and recommendation generation. The experimentation has been done on MovieLens dataset used by GroupLens Movie Recommender System. Recommendations are generated by taking the average of frequency count of rating 1, 2,3,4,5 by different users having demographic similarity.

The prediction accuracy is measured using Mean Absolute Error, Root Mean Squared Error, Precision, Recall and F-score. This paper concludes that demographic data in the MovieLens dataset does not influence differently on users ratings. For future research, the results can be enhanced by creating more than one training and testing dataset to be evaluated and gather the average of the results. Recommendation quality can be improved by relating the movie genres to demographic attributes or we can apply the same framework on different problem domains datasets.

In [12] hybrid recommender system approach is used which is based upon collaborative filtering and demographic filtering to solve user-cold start problem. First of all users of similar preferences are grouped together in a cluster using user-item rating matrix and then demographic information of the users is combined with clusters information. Now, clusters are classified by building decision tree and it generates some rules. These rules help when a new user arrives in a system by identifying a cluster to which he/she belongs. At the end recommendations are presented on the basis of his/her cluster member's preferences.

Time required to do offline analysis is O(mn) where m is the no. of users and n is the no. of items. The time required to find cluster for new user is O(1) and time to calculate average rating for new user on the basis of users preferences in the same cluster to which he belongs is O(n). This method reduced the online cost for the overall recommendation process by identifying similar users in less time complexity as compared to other methods under experimentation such as user-based CF, community based filtering and so on. The only limitation in this method is dependency upon user's registration information for generation of accurate recommendations. In this paper suggestion for future is to use the same methodology with more no. of demographic attributes and experimentation can be extended to solve item-cold start problem and large dataset can be used for better analysis of recommender system evaluation.

In [13] the concept of social-tags is used to design recommender systems which are mainly focused on identifying community. Such kind of systems can be described as a network where nodes represent individuals, and edges denote the relations among them. In this approach, users are allowed to freely assign tags to annotate their collections and as a result strengthen the semantic relations among users and objects. In this paper, a diffusion-based recommendation algorithm is used which treats tags as

a means of connecting users and objects. In this usage frequency of tags is considered as user's personal preference and semantic relations between tags and objects as global information. In this entropy-based and Hamming-distance-based methods are employed to measure the inner and inter-diversity of tag usage patterns. The proposed algorithm using user-tag-object diffusion in this paper results in improvement of recommendation accuracy in cold-start problem. Experiments have shown different tag usage patterns in two datasets, users assign more diverse tags in Del.icio.us dataset than MovieLens dataset. The proposed algorithm is tested against metrics: Ranking Score, Inter Diversity and Inner Diversity. Thee polysemy and synonymy problem might result in coarse and inaccurate performance of recommender systems based on social tags.

In [14] experimentation is done using three type of relationships i.e. user-user similarity, wine-wine similarity, user preference relationship. This paper dealt with many problems of recommender system like Data Sparsity, Grey sheep and Synonymy problem including cold-start problem. The dataset for experiments is taken from three wine-based vendor websites eBacchus.com, wine.com and tastings. These websites contains wine tasting notes which is further parsed using Stanford's part-of-speech tagging tool for extracting attributes phrases based on lexical patterns like adjective-adjective. Through this parsing process we get various wine attributes for populating the dataset for testing purpose. Out of all, top 25 attributes are selected based upon their frequency count for using as classification tags. This dataset is further used for finding degree of wine-wine similarity using Jaccard's Coefficient for generating top-N recommendations. Similarly, user-user similarity is calculated for identifying clusters of users with same preferences of wine. The proposed methodology in paper is based upon combination of content- based and collaborative filtering based approach for finding recommendations for target user. In [14] main focus is given on establishing user trust on recommendations produced by their proposed algorithm. This approach deals with cold-start problem when user is having few preferences for wine at initial stage. Further the list is expanded after finding similarity between user preferences and existing wine information available in dataset. Here top k1, k2 model is used for recommendation generation. Top k1 user preferences are considered along with top k2 wines with the highest wine-wine similarity index scores. So, the total no. of recommendations are k1 * k2. The recommendations are evaluated using RMSE metric and the only limitation here is

that we only have limited user preferences available to solve user cold-start problem. The quality of recommendation of wine can be improved in future using more no. of wine attributes list or some other ranking algorithm can be used for generating top-ranked wines.

In [15] a study is made using MovieLens dataset that whether online recommenders systems can perform better than recommendations by Human Beings. This interesting study focuses on finding how humans recommends in different situations and same approach can be applied for improving the recommendations generated by any online recommender system. The results from MovieLens algorithmic predictions are compared with MovieLens active online users recommendations. This paper gives new insights into why and where recommender system lags. This paper gives us a great scope for improving recommender system generated recommendations to fit into user preferences.

In [16] recommendation generation based upon community detection using different dimensions of social networks. These dimensions refer to friendship network, item-based similarity network and commenting network. Community detection in social media or network analysis helps in understanding collective user behaviors more appropriately. This concept can also be used for group commendations rather than just recommendations for an individual. For this purpose various graph based methods are used for finding network among users on social network websites. Further it is assumed that interactions are more among groups or communities rather than outside group. So, this fact can be used for finding similarities among group members for finding their preferences. This paper is based upon modularity based community detection method for multi-dimensional social networks. Here, new user cold-start problem is solved by assuming that if he is not having any history in this system but have user profile details in another system then that external profile information is considered for finding relevant recommendations for this user in current system. The community detection based recommendation generation is more effective as compared to producing recommendations using entire user database. In future different community detection algorithms and approaches can be used and community size can be varied for testing recommendation quality.

Chapter 3

Problem Statement

3.1 Introduction

User-based Collaborative-filtering (CF) technique is the most frequently used recommender technique due to its simplicity and efficient performance. With this technique system is capable of finding neighbourhood i.e. users with similar taste or preferences. The recommendations are generated from this group based preferences but user-based CF approach suffers from limitation of finding top-N recommendation if user-item matrix is **sparse**. Another alternative is to find recommendation using item-based similarity which is referred to as content-based filtering. But both the approaches are incapable of predicting preferences of new user in a system known as user-cold start problem or new item cold-start problem.

So, a new approach referred as hybrid recommender systems is used here in this research work to address cold-start problem of recommender systems. Many approaches are used by researchers till now to solve this issue as discussed in literature review but still there is a lot of scope for further improvement in generating quality recommendations for a new user.

3.2 OBJECTIVES

The main focus of thesis work is to deal with cold-start problem using demographic information based user-similarity. The demographic information is composed of various user characteristics like Age, Gender, Occupation, Zip-Code, race, religion, marital-status, locality, hobbies etc. This approach is preferred when user rating history is not available for generating preferences. So, this work is effective for finding recommendations using external information provided by users at registration time. The main objectives to be achieved through this work are listed below:

3.2.1 OFFLINE ANALYSIS OBJECTIVES

- To prepare a refined dataset specific to problem domain for testing purpose.
- To achieve appropriate clustering of instances in unsupervised machine learning scenario when no labels or classes are provided to instances.

- To analyse the performance of clustering by classification.
- To find combination of demographic attributes for accurate classification of instances and for generating decision rules.

3.2.2 ONLINE ANALYSIS OBJECTIVES

- To find neighbourhood consisting of similar users to new user who has just arrived in the system when insufficient rating data or no rating data is available.
- To predict top-N recommendations on the basis of users in neighbourhood using database integration.
- To present the recommendation to the new user online with less time complexity.

Chapter 4

Proposed Methodology and Technologies used

4.1 Introduction to Methodology

The proposed framework for finding recommendation for a new user cold-start problem is based upon Hybrid recommender system. The main data mining techniques used here are K-means clustering algorithm for finding user similarity based upon users demographic characteristics like UserID, Age, Gender and Occupation and then evaluating clusters performance using J48 classification algorithm. AddCluster filtering algorithm is used for assigning clusters to instances in user dataset because of unavailability of class labels for instances. It is a form of unsupervised machine learning. The proposed approach is divided into two parts:

- Offline analysis using Weka tool for building a model for collaborative filtering using appropriate demographic attributes.
- Fetching new user data online and then followed by finding similar users with a new user. Finally recommendations are generated for new user using existing similar users rating data.

Data pre-processing is considered as initial phase in any data mining task. Clustering and Classification are two most popular data mining techniques for finding hidden patterns in a given dataset. In general, classification technique is used when we already have knowledge about to which class a particular instance or object belongs. On the other hand, Clustering is performed on the basis of identifying similarity among attribute values defining a particular object and then grouping is done for similar objects. In machine learning terminology, classification is called as *supervised learning* and clustering is referred to as *unsupervised learning*.

Clustering methods is of three types: partitioning methods, density-based methods, and hierarchical methods. K-means, proposed by MacQueen is an example of most-commonly used partitioning method due to two major advantages i.e. relative efficiency and ease of implementation. Density-based clustering methods are used for searching dense clusters of objects which are separated by sparse regions that

represent noise for example; *DBSCAN* and *OPTICS*. Hierarchical clustering methods, decompose a set of data objects in a particular hierarchy based upon some criteria for example; *BIRCH* [5].

Clustering models have better scalability than typical collaborative filtering methods because they make predictions within much smaller clusters rather than the entire customer base. The complex and expensive clustering computation is run offline.

4.2 K-Means clustering Algorithm for Collaborative Filtering

An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing N_j data points so as to minimize the sum-of-squares criterion [11]

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where x_n is a vector representing the nth data point and μ_j is the geometric centroid of the data points in S_j. In general, the algorithm does not achieve a global minimum of J over the assignments. In fact, since the algorithm uses discrete assignment rather than a set of continuous parameters, the "minimum" it reaches cannot even be properly called a local minimum. Despite these limitations, the algorithm is used fairly frequently as a result of its ease of implementation.

The algorithm consists of a simple re-estimation procedure as follows. Initially, the data points are assigned at random to the K sets. For step 1, the centroid is computed for each set. In step 2, every point is assigned to the cluster whose centroid is closest to that point. These two steps are alternated until a stopping criterion is met, i.e., when there is no further change in the assignment of the data points.

A global minimum, also known as an absolute minimum, is the smallest overall value of a set, function, etc., over its entire range. It is impossible to construct an algorithm that will find a global minimum for an arbitrary function.

A local minimum, also called a relative minimum, is a minimum within some neighbourhood that need not be (but may be) a global minimum.

4.3 Architecture for proposed approach

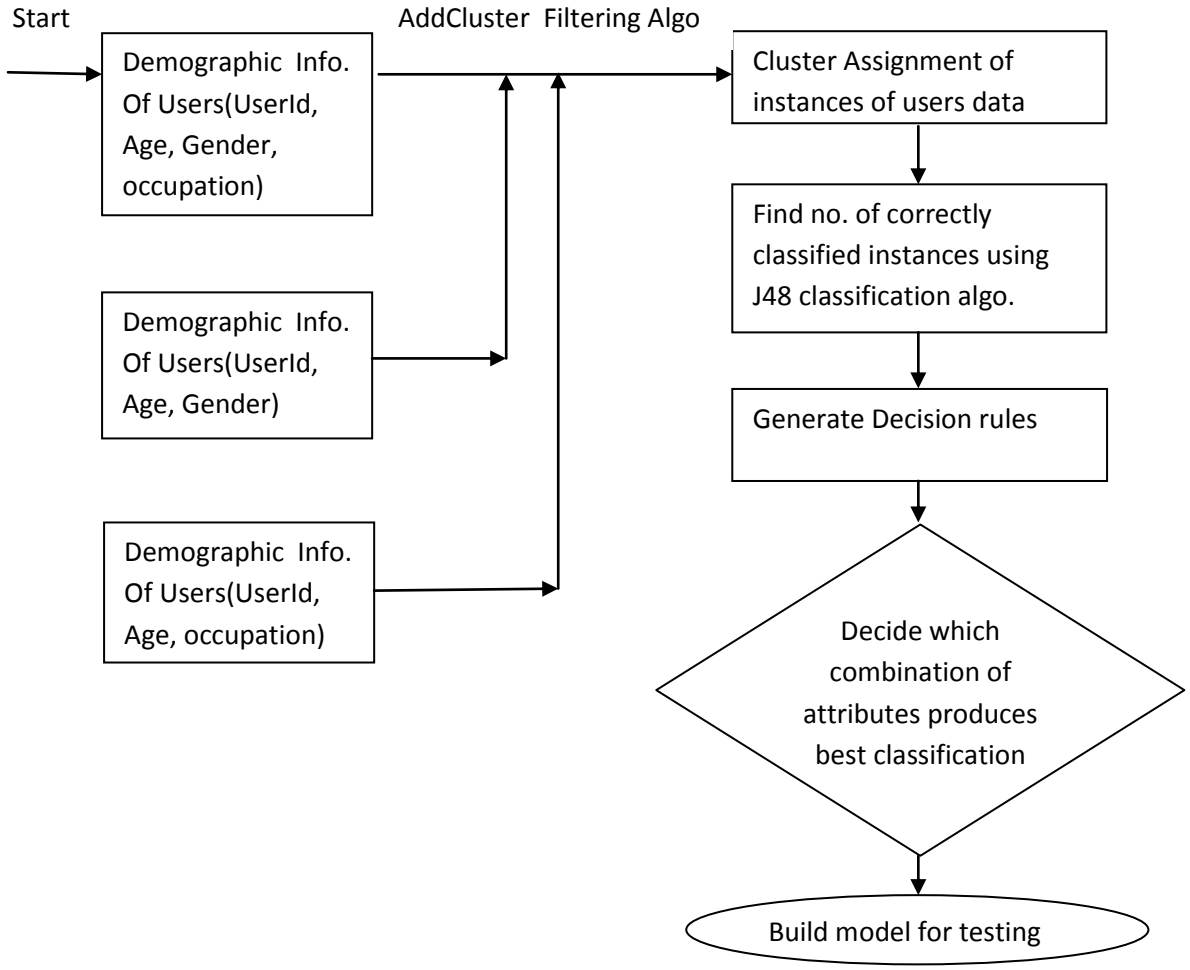

Figure 7: Offline Analysis using Weka

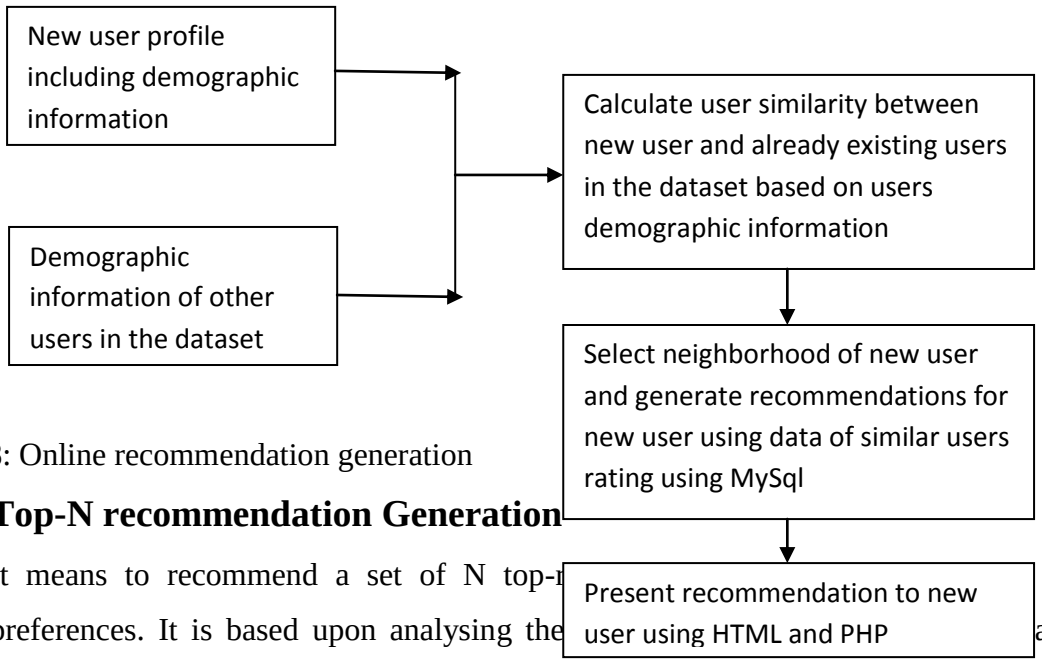

Figure 8: Online recommendation generation

4.4 Top-N recommendation Generation

It means to recommend a set of N top-r user preferences. It is based upon analysing the ation

between different users or items for computing the recommendations. Top-N recommendations are further generated using either user-based or item-based Top-N recommendations [5].

In user-based Top-N recommendations algorithm first of all find neighbourhood of size K i.e. similar users in interest with active user and then corresponding to rows in user-item matrix aggregate analysis is performed to know which are most liked by group of those similar users as per frequency count of items. The recommendations generated through this approach have limitation related to scalability and real-time performance.

Item-based Top-N recommendation approach is used to solve the limitations of user-based CF for generating Top-N recommendations. Here, the focus is shifted from user-similarity to Item-similarity computation for each item. The result will be set of items say C, which can be a candidate for recommendations to the user. Now, those items will be removed which the user has already purchased and replaced with Top-N recommended items which are similar to items that the user has already purchased. This approach fails when the user has no rating history or he/she has not purchased any item till now. Now, the only solution left is to explore the possibility of finding items for recommendation on the basis of users interests explicitly i.e. by using external information from user profiles containing demographic data. In proposed approach recommendations are generated by using MySql database created for MovieLens dataset and then respective queries are run to find required recommendation for new user. By using Sampling or by reducing dimensions in user-item matrix, clustering can be performed optimally over large data sets but it may affect recommendation quality to some extent.

4.5 TECHNOLOGIES USED

The two main technologies used in this thesis work are Weka Tool for Data Mining and MySql for generating recommendations.

4.5.1 Weka Data Mining Software

Weka is extensively used by data mining and machine learning research communities, since 1994. It is preferred for doing offline analysis in this thesis work because of following key features: [12]

- It incorporates plenty of data mining and machine learning algorithms for building models.
- It is a open source tool and available free of cost.
- It is platform-independent and easily installable on all operating systems like Windows, Mac and Linux etc.
- It is user friendly as help is provided along with every algorithm for better understanding and that's the reason why even non data mining experts prefer this tool for research and experimentation.
- It can be easily integrated with languages like java.
- Weka has been kept up-to-date, by adding new algorithms of data mining and machine learning as soon as they found importance in research literature.
- Lots of learning resources including video tutorials are freely available online.

4.5.2 MySql Database

MySql is used for running queries for generating recommendations for new user in a system. The MovieLens database is created used this database as it is easily managed using GUI based phpmyadmin environment.

It is open-source database and we need xampp installation before running any query in it. The learning of this is quite easy due to availability of many online resources which proved to be a great source of help in completing experimentation work. It can be easily integrated with web development scripting languages like PHP for producing recommendation results in a presentable and organized manner. It provides powerful features like database creation, consistency and integrity along with security features to protect hacking of confidential data of online users of recommender system.

Chapter 5

Introduction to Dataset and Experiment Results

5.1 Introduction to MovieLens Dataset

Dataset Chosen: ml-latest, a Movie database [10]

This dataset (ml-latest) describes 5-star rating activity from [MovieLens](http://movielens.org), a movie recommendation service. It contains 21063128 ratings and 470509 tag applications across 27303 movies. These data were created by 229060 users between January 09, 1995 and March 31, 2015. This dataset was generated on March 31, 2015.

Users were selected at random for inclusion. All selected users had rated at least 1 movie. The data for experimentation is taken from three files, `users.csv`(information taken from older version of movies database and converted into .csv file for experimentation), `movies.csv`, `ratings.csv` . The dataset files are written as [comma-separated values] files with a single header row. Columns that contain commas (`,`) are escaped using double-quotes (`"`). These files are encoded as UTF-8.

Movie ids are consistent between `ratings.csv`, `movies.csv`, and `users.csv` (i.e., the same id refers to the same movie across these three data files). For experimentation a sample of 6040 users is taken.

DATA FILES DETAILS OF MOVIELENS DATABASE

i) **Ratings Data File Structure (ratings.csv)**

This file contains information in the format given below:

userId, movieId, rating, timestamp

The records are ordered first by userId, then, within user, by movieId. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

ii) **Movies Data File Structure (movies.csv)**

Movie information is contained in the file `movies.csv`. This file has the following format:

movieId, title, genres

Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles. Genres are a pipe-separated list, and are selected from the following:

Action| Adventure| Animation| Children's| Comedy| Crime| Documentary| Drama| Fantasy|Film-Noir| Horror| Musical| Mystery| Romance| Sci-Fi| Thriller| War| Western| (no genres listed)

For experiment purpose the genres are separated into different columns and if a particular movieId belongs to a particular genre then '1' is replaced below that column in .csv file else '0' is replaced below that column.

iii) USERS DATA FILE STRUCTURE

User information is in the file "users.dat" and is in the following format:

UserID::Gender::Age::Occupation::Zip-code

This file is converted into .csv for experimentation using weka tool. All demographic information is provided voluntarily by the users and its detailed description is shown below:

- Gender is denoted by **"M"** for male and **"F"** for female

- Age is chosen from the following ranges:

* 1: "Under 18"
* 18: "18-24"
* 25: "25-34"
* 35: "35-44"
* 45: "45-49"
* 50: "50-55"
* 56: "56+"

- Occupation is chosen from the following choices:

0: "other" or not specified
1: "academic/educator"
2: "artist"
3: "clerical/admin"
4: "college/grad student"
5: "customer service"
6: "doctor/health care"
7: "executive/managerial"
8: "farmer"
11: "lawyer"
12: "programmer"
13: "retired"
14: "sales/marketing"
15: "scientist"
16: "self-employed"
17: "technician/engineer"
18: "tradesman/craftsman"
19: "unemployed"

9: "homemaker" 20: "writer"

10: "K-12 student"

The experiment focuses on finding how clustering results can be achieved well based upon appropriate choice of attributes using k-means clustering algorithm approach and further clusters performance is validated using J48 Classification algorithm. After experimentation in Weka tool for data mining, a model is proposed for further testing and for retrieving recommendation results using MySQL.

Note: Tool Used for experimentation: Weka Tool and MySQL [19] [20] [21]

5.2 Experiment I: PART-A

In Part-A of experiment, cluster assignments is depicted for every instance. The selected filter for knowing the cluster assignments is **AddCluster** which makes use of k-means clustering algorithm for assigning clusters to each instance in the dataset and the end result is information containing nominal attribute added at the end of each record showing to which cluster each instance belongs. Zip-code attribute is removed before applying the filter to data.

Procedure to run AddCluster filter [19]

1. Click on choose button for selecting filter (Weka → filter → unsupervised learning → attribute → AddCluster)

   Options to set for AddCluster filtering algorithm

   - clusterer -- The clusterer to assign clusters with (here simple k-means algorithm isI used for clustering and we need to set no. of clusters for this algo.)
   - ignoredAttributeIndices -- The range of attributes to be ignored by the clusterer. eg: first-3,5,9-last

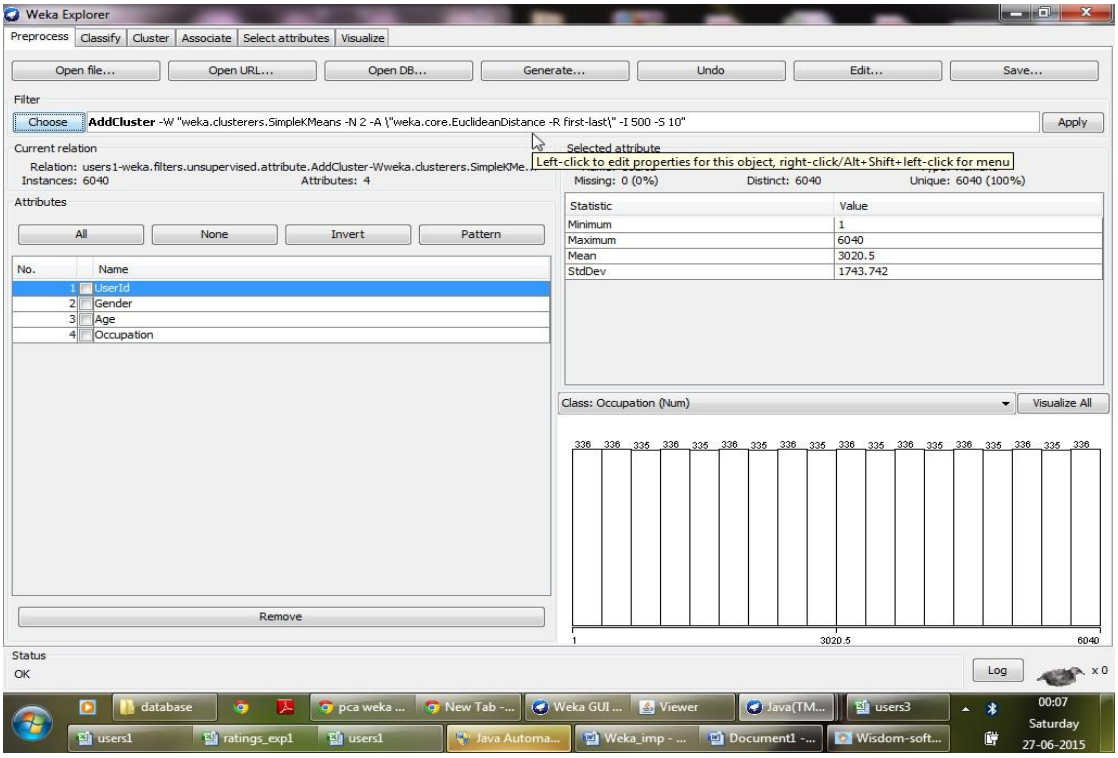

Figure 9: AddCluster filtering algorithm using Weka tool for Experiment I

2. Addcluster filter makes use of k-means algorithm and select numClusters property to 4 before clicking on apply button. Right-click on AddCluster filter name for setting its properties and click on more button to know about AddCluster filter details as shown below:

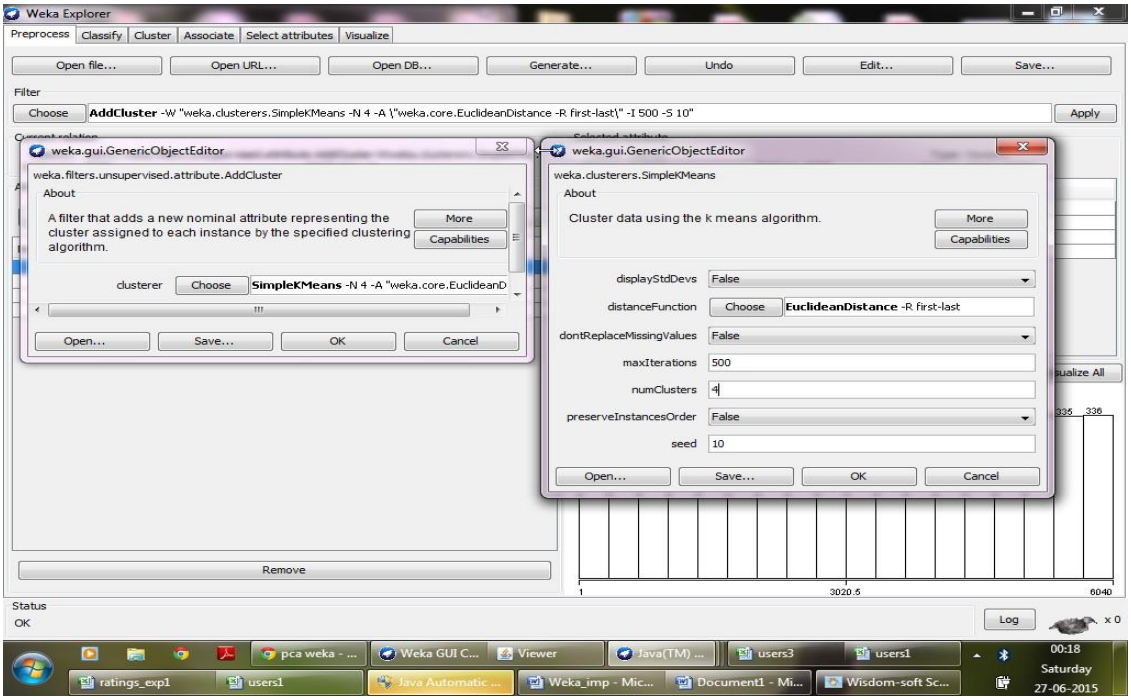

Figure 10: Settings for AddCluster filtering algorithm for Experiment I

3. Click on apply button to see the effect of AddCluster filter. Now click on Edit Button to view the cluster assignments.

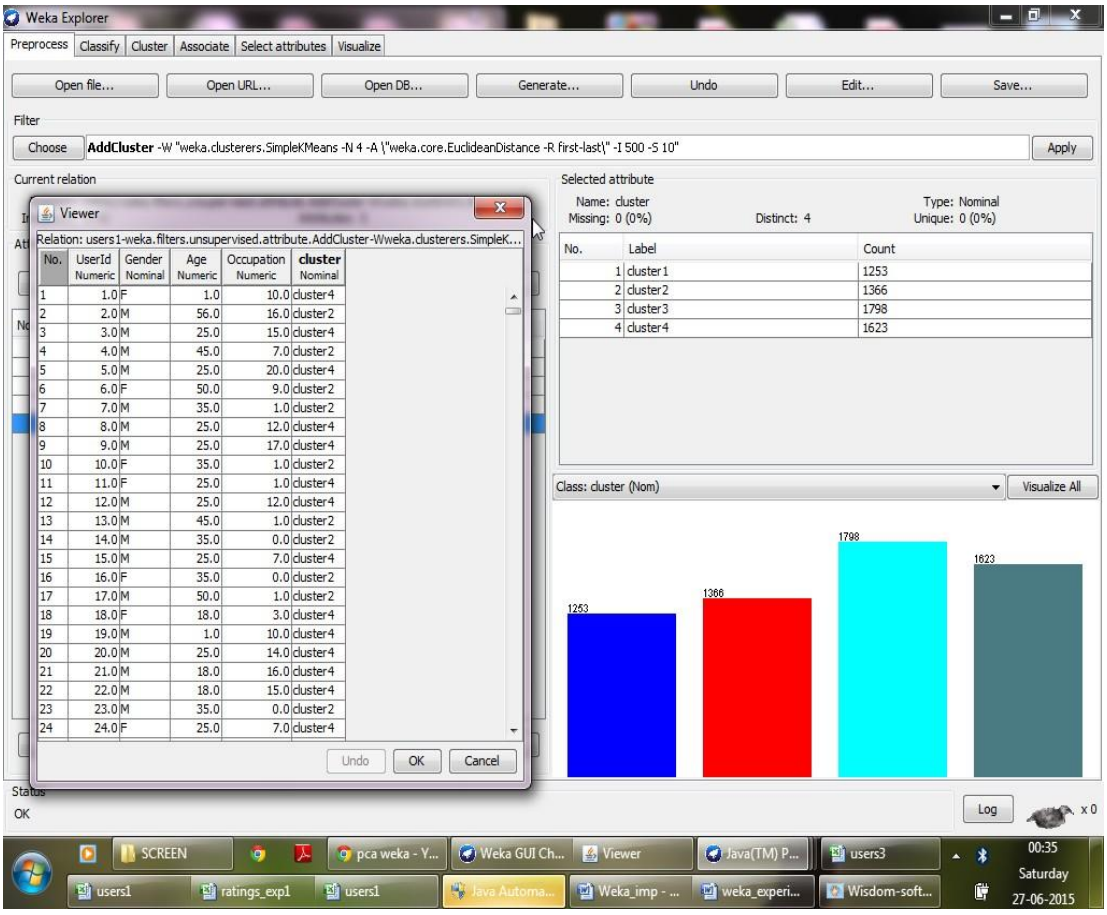

Figure 11: View of cluster assignment for each individual instance for Experiment I

5.3 Experiment I: PART-B [20]

Now use J48 classification algorithm to evaluate the performance of the clustering algorithms with **Use cross-fold validation option** [24]**.** In k-fold validation the no. of instances in dataset is partitioned into k folds or sets of approximately same size. The evaluation process is done k times and every time one fold is considered for testing and remaining folds for training purpose. In this approach each instance is taking part in testing at least once [2].

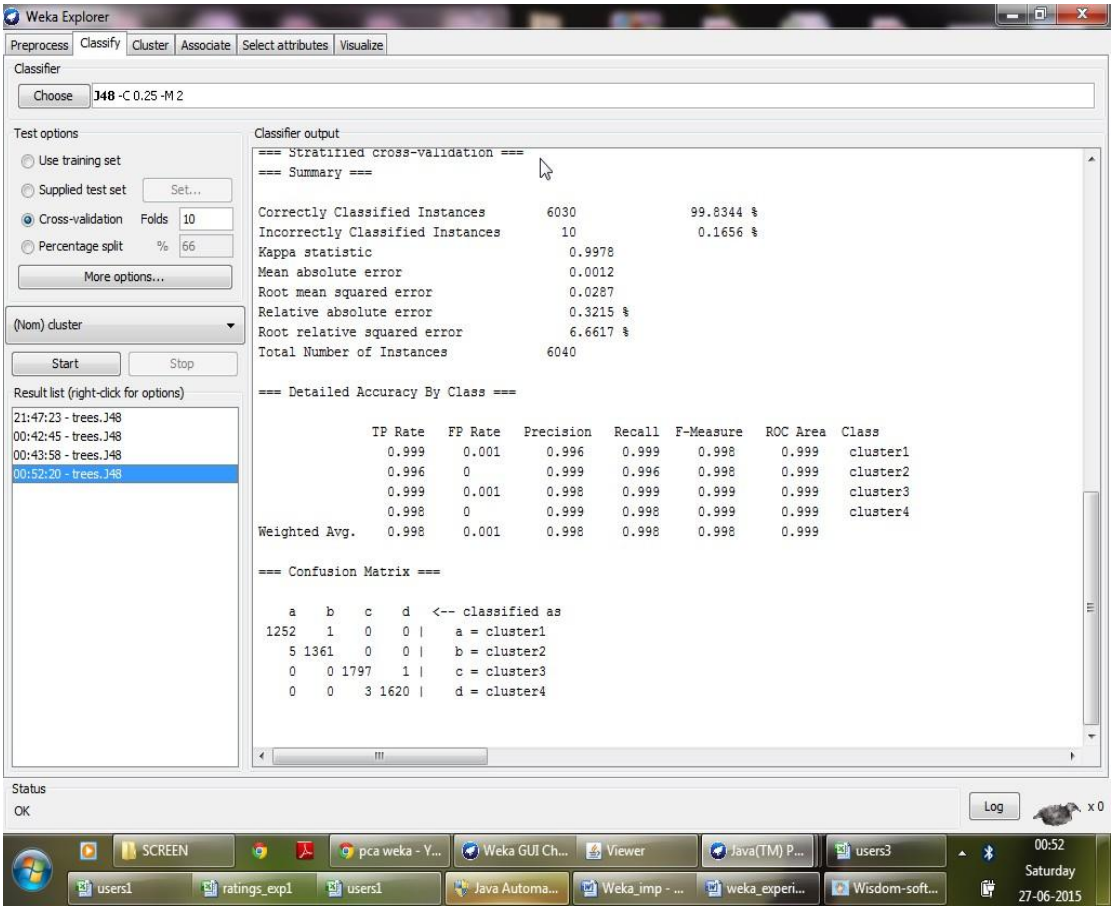

Figure 12: Result view of J48 classification algorithm for Experiment I

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	**6030**	**99.8344 %**
Incorrectly Classified Instances	**10**	**0.1656 %**
Kappa statistic	0.9978	
Mean absolute error	0.0012	
Root mean squared error	0.0287	
Relative absolute error	0.3215 %	
Root relative squared error	6.6617 %	
Total Number of Instances	6040	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.999	0.001	0.996	0.999	0.998	0.999	cluster1
0.996	0	0.999	0.996	0.998	0.999	cluster2
0.999	0.001	0.998	0.999	0.999	0.999	cluster3
0.998	0	0.999	0.998	0.999	0.999	cluster4

Weighted Avg. 0.998 0.001 0.998 0.998 0.998 0.999

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1252	1	0	0	a = cluster1
5	1361	0	0	b = cluster2
0	0	1797	1	c = cluster3
0	0	3	1620	d = cluster4

5.4 Experiment II: PART-A

Repeat the same procedure with less no. Of attributes i.e. **UserId, Gender, Age** and ignore rest of the attributes in the table.

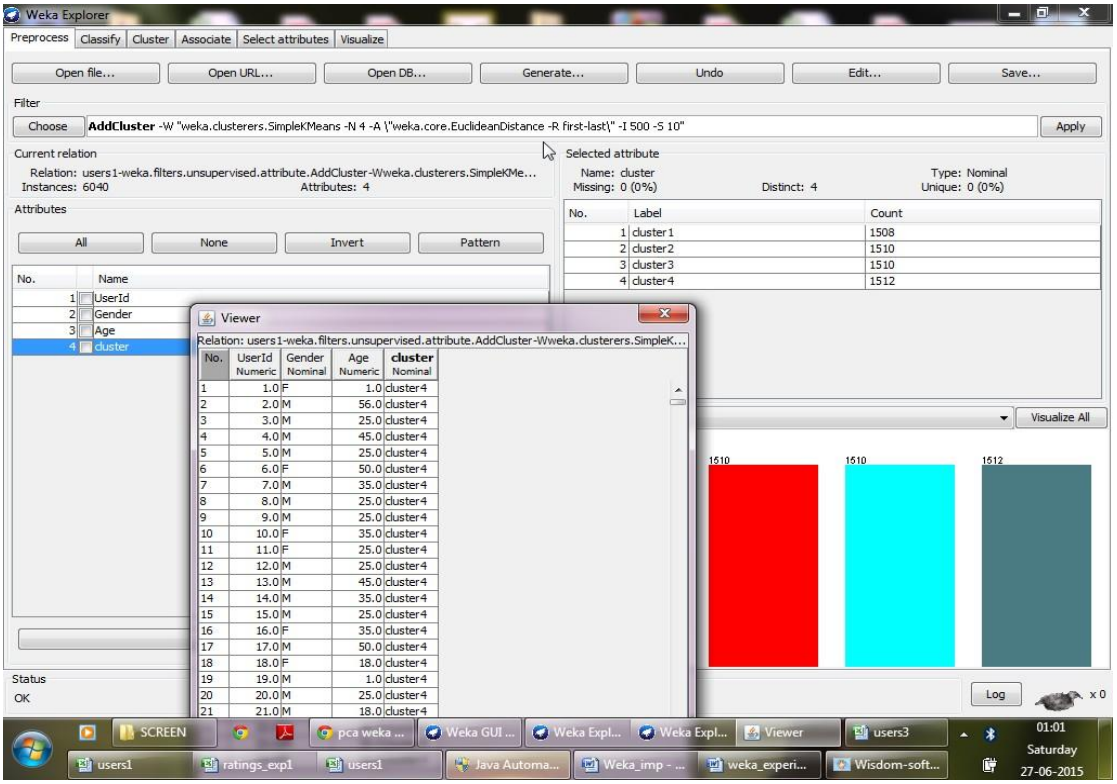

Figure 13: View of cluster assignment for each individual instance for Experiment II

5.5 Experiment II: PART-B

Now again use J48 classification algorithm to evaluate the performance of the clustering algorithms with Use cross-fold validation.

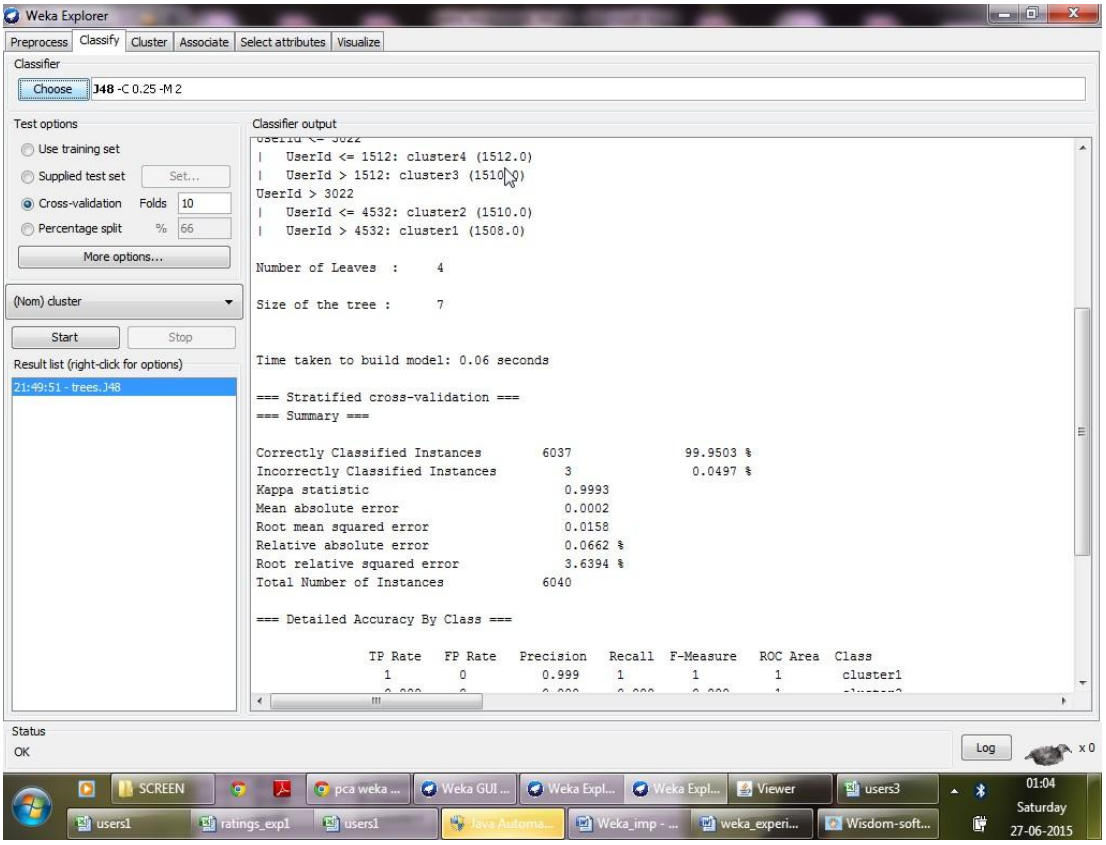

Figure 14: Result view of J48 classification algorithm for Experiment II

=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: users1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10-weka.filters.unsupervised.attribute.Remove-R5-weka.filters.unsupervised.attribute.Remove-R4-5-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Instances: 6040
Attributes: 4
UserId
Gender
Age
cluster
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

UserId <= 3022
| UserId <= 1512: cluster4 (1512.0)
| UserId > 1512: cluster3 (1510.0)
UserId > 3022
| UserId <= 4532: cluster2 (1510.0)
| UserId > 4532: cluster1 (1508.0)

Number of Leaves : 4
Size of the tree : 7
Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	**6037**	**99.9503 %**
Incorrectly Classified Instances	**3**	**0.0497 %**
Kappa statistic	0.9993	
Mean absolute error	0.0002	
Root mean squared error	0.0158	
Relative absolute error	0.0662 %	
Root relative squared error	3.6394 %	
Total Number of Instances	6040	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	0.999	1	1	1	cluster1
	0.999	0	0.999	0.999	0.999	1	cluster2
	0.999	0	0.999	0.999	0.999	1	cluster3
	0.999	0	1	0.999	1	1	cluster4
Weighted Avg.	1	0	1	1	1	1	

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1508	0	0	0	a = cluster1
1	1509	0	0	b = cluster2
0	1	1509	0	c = cluster3
0	0	1	1511	d = cluster4

5.6 Experiment III: PART-A

Repeat the same procedure with less no. of attributes i.e. UserId, Age, Occupation

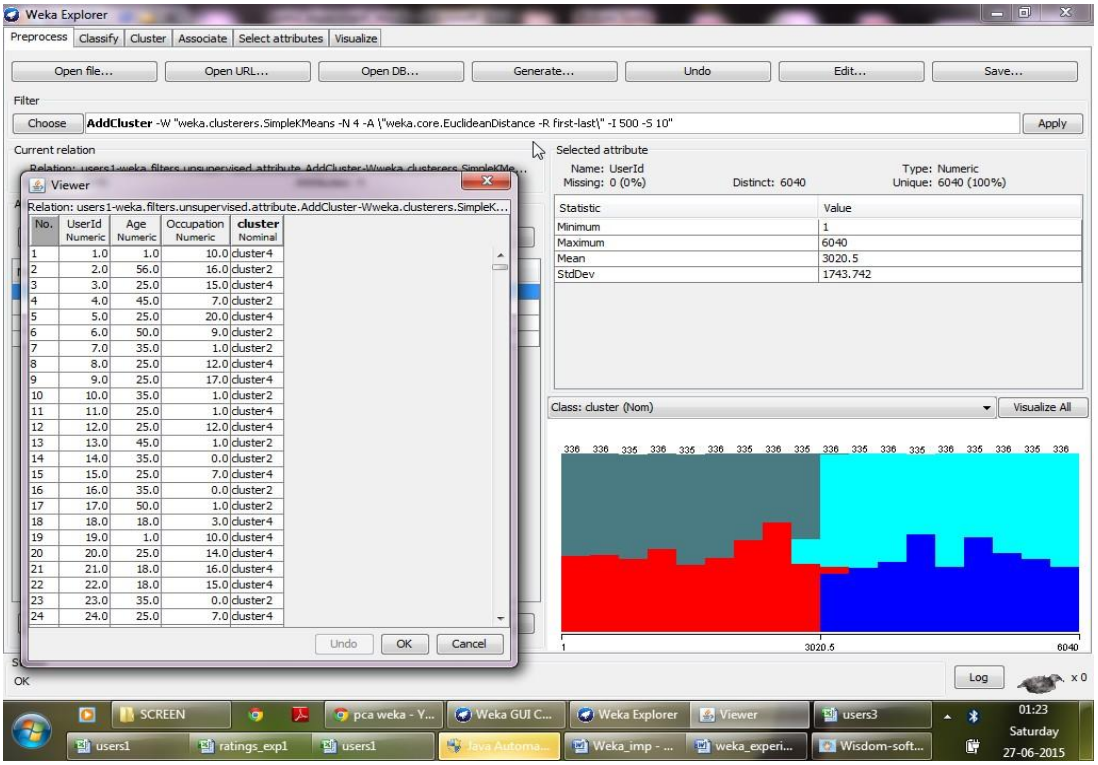

Figure 15: View of cluster assignment for each individual instance for Experiment III

5.7 Experiment III: PART-B

Now again use J48 classification algorithm to evaluate the performance of the clustering algorithms with Use cross-fold validation.

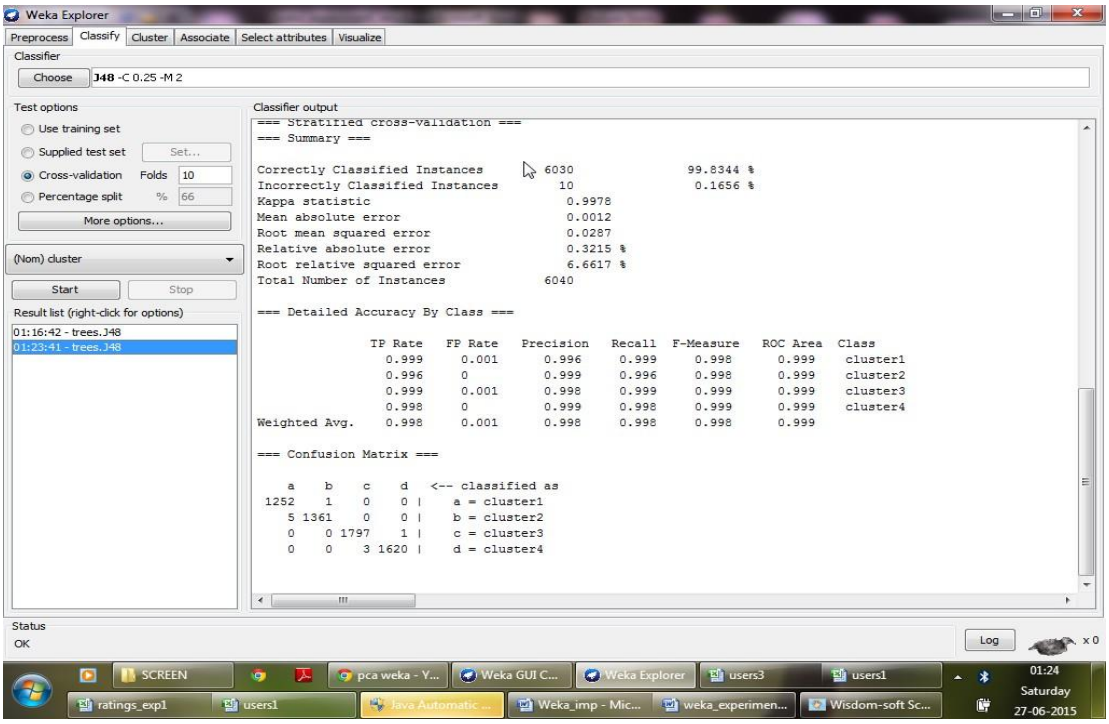

Figure 16: Result view of J48 classification algorithm for Experiment III

=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: users1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10-weka.filters.unsupervised.attribute.Remove-R5-weka.filters.unsupervised.attribute.Remove-R2,5-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10-weka.filters.unsupervised.attribute.Remove-R4-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.SimpleKMeans -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Instances: 6040
Attributes: 4
UserId
Age
Occupation
cluster
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

Age <= 25
| UserId <= 2937: cluster4 (1615.0/1.0)
| UserId > 2937
| | UserId <= 3046
| | | Age <= 18
| | | | UserId <= 2965: cluster4 (7.0)
| | | | UserId > 2965
| | | | | Age <= 1: cluster4 (2.0)
| | | | | Age > 1: cluster3 (26.0)
| | | Age > 18: cluster3 (33.0)
| | UserId > 3046: cluster3 (1738.0)
Age > 25
| UserId <= 3049: cluster2 (1364.0/1.0)
| UserId > 3049: cluster1 (1255.0/3.0)

Number of Leaves : 8
Size of the tree : 15
Time taken to build model: 0.05 seconds
=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	**6030**	**99.8344 %**
Incorrectly Classified Instances	**10**	**0.1656 %**
Kappa statistic	0.9978	
Mean absolute error	0.0012	
Root mean squared error	0.0287	
Relative absolute error	0.3215 %	
Root relative squared error	6.6617 %	
Total Number of Instances	6040	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.999	0.001	0.996	0.999	0.998	0.999	cluster1
0.996	0	0.999	0.996	0.998	0.999	cluster2
0.999	0.001	0.998	0.999	0.999	0.999	cluster3
0.998	0	0.999	0.998	0.999	0.999	cluster4
Weighted Avg. 0.998	0.001	0.998	0.998	0.998	0.999	

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1252	1	0	0	a = cluster1
5	1361	0	0	b = cluster2
0	0	1797	1	c = cluster3
0	0	3	1620	d = cluster4

Table 3: Comparison statistics of clustering on users.csv table

S.No.	**J48 classifier Using 4 attributes** (UserId, Age, Gender, Occupation)	**J48 classifier Using 3 attributes** (UserId, Age, Gender)	**J48 classifier Using 3 attributes** (UserId, Age, Occupation)
1.	Correctly Classified Instances 6030 (99.8344 %) Incorrectly Classified Instances 10 (0.1656 %)	Correctly Classified Instances 6037 (99.9503 %) Incorrectly Classified Instances 3 (0.0497 %)	Correctly Classified Instances 6030 (99.8344 %) Incorrectly Classified Instances 10 (0.1656 %)

The classification based upon **J48 classifier Using 3 attributes** (UserId, Age, Gender) is best in terms of correctly classified instances. So, UserId, Age and Gender proves to be best attributes for clustering or for finding user similarity for predicting recommendations for new user in the system who is not having any rating history.

5.8 Solving Cold Start Problem using the user similarity based upon AddCluster filter (using k-means algorithm)

1. First of all run the AddCluster filter algorithm with actual records in the data file to check the clusters for all the instances.

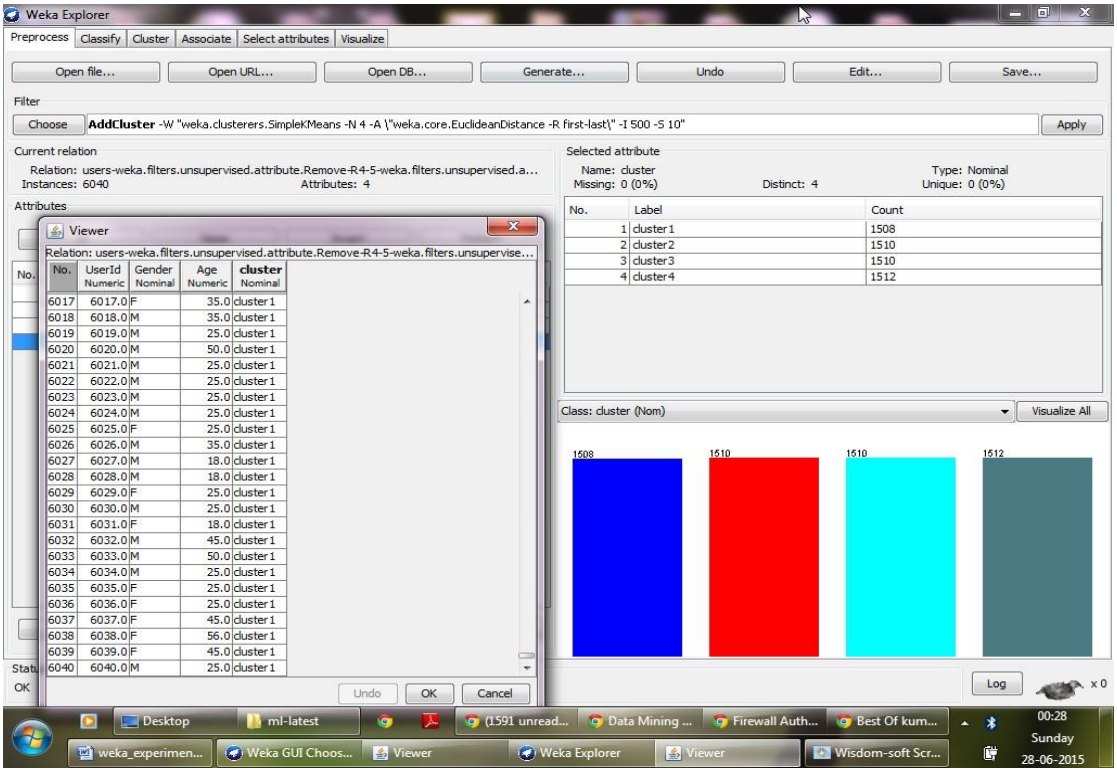

Figure 17: View of cluster assignment based on UserId, Gender, Age

2. Now suppose a new user enters into the system with user id 6041 and the same algorithm is applied to know to which the new user belongs. The new user belongs to **cluster 3** as shown in the screenshot below.

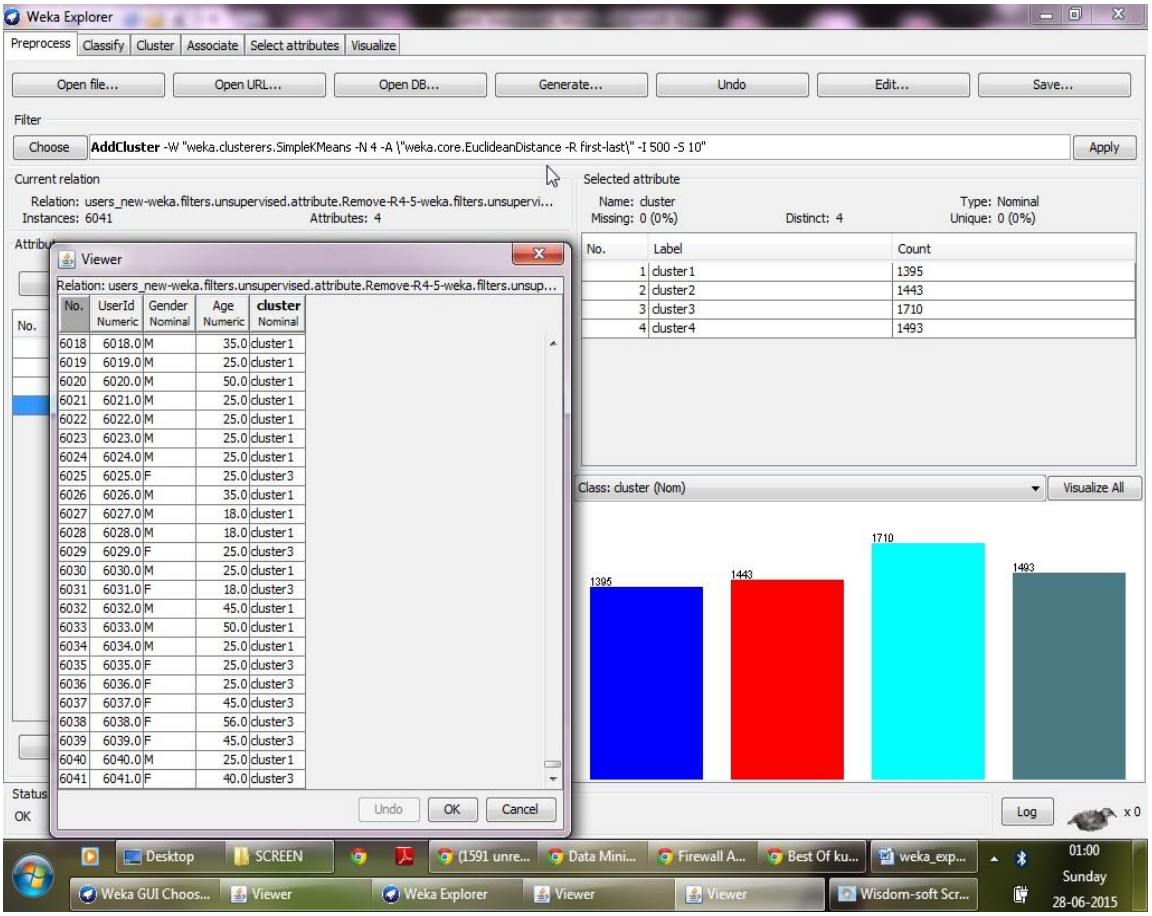

Figure 18: View of cluster assignment based on UserId, Gender, Age for new user with userId 6041

3. Now we can predict the recommendations for these user based upon his/her cluster members ratings.

5.9 Recommendation Generation using MySQL

Choose UserId, Age, Gender as the three attributes for finding user similarity based upon k-means algorithm as best way for clustering the User Demographic information dataset. From now onwards the experimentation makes use of Mysql to carry out various queries to know top10 movie recommendation for the new user in the system.

The sample dataset is consisted of 500 records from users.csv file and saved as user2.csv file. Another refinement is done on movie.csv dataset consisted of 1200 movies. The ratings.csv file contains user rating records corresponding to movie ratings given by 500 users on 1200 movies and rest of the records are not considered. The tables under experimentation are imported using PhpMyAdmin and under a new Mysql database with the name **dbmovielens**.

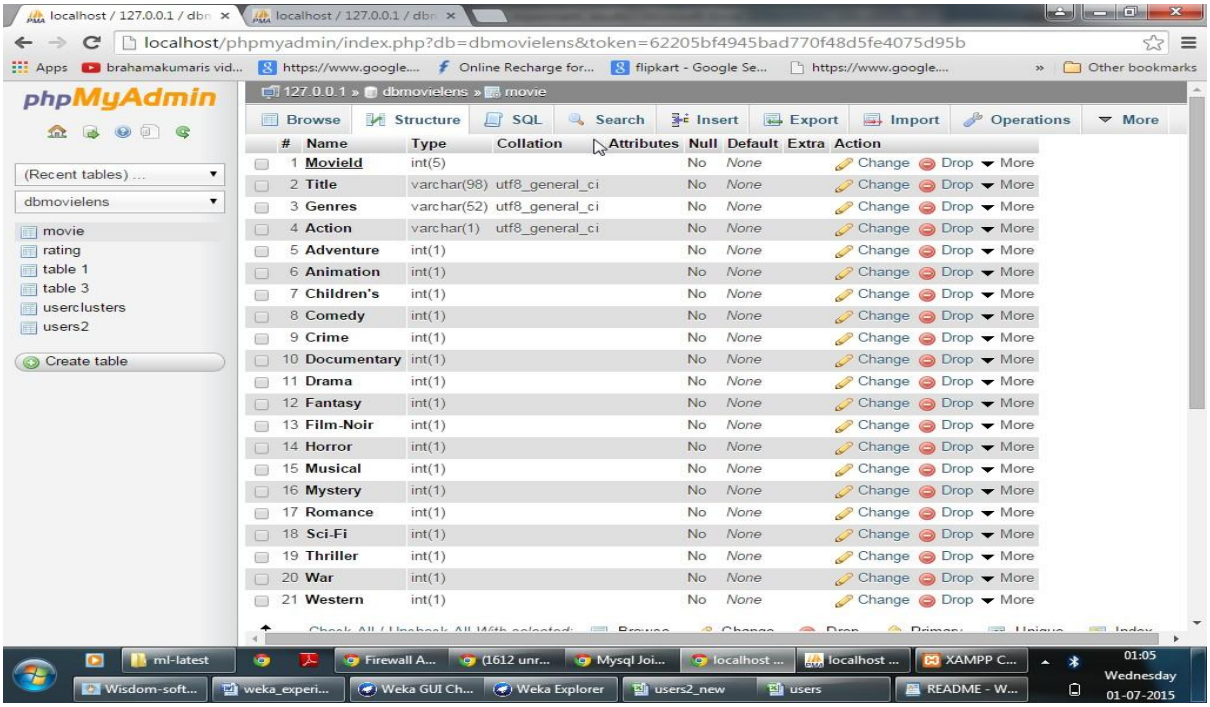

Figure 19: Schema of movie data table

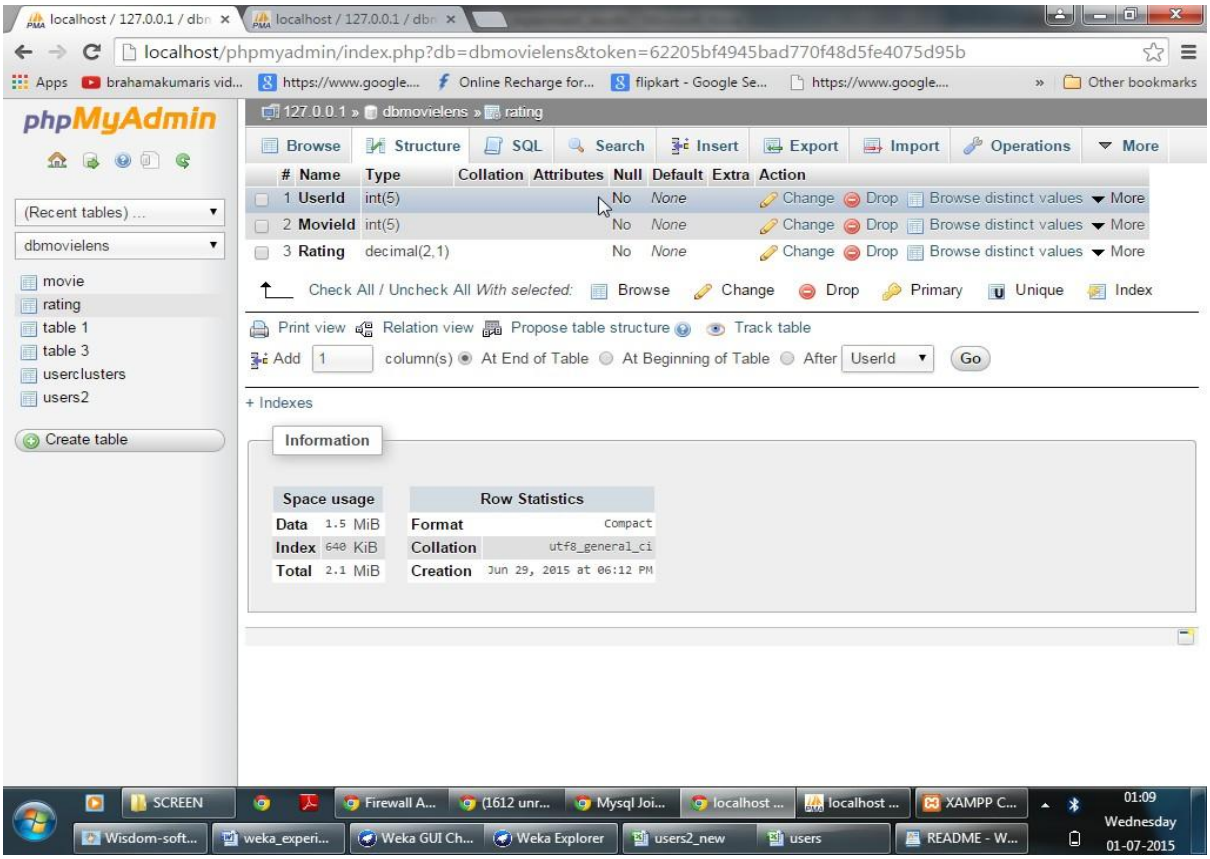

Figure 20: Schema of rating data table

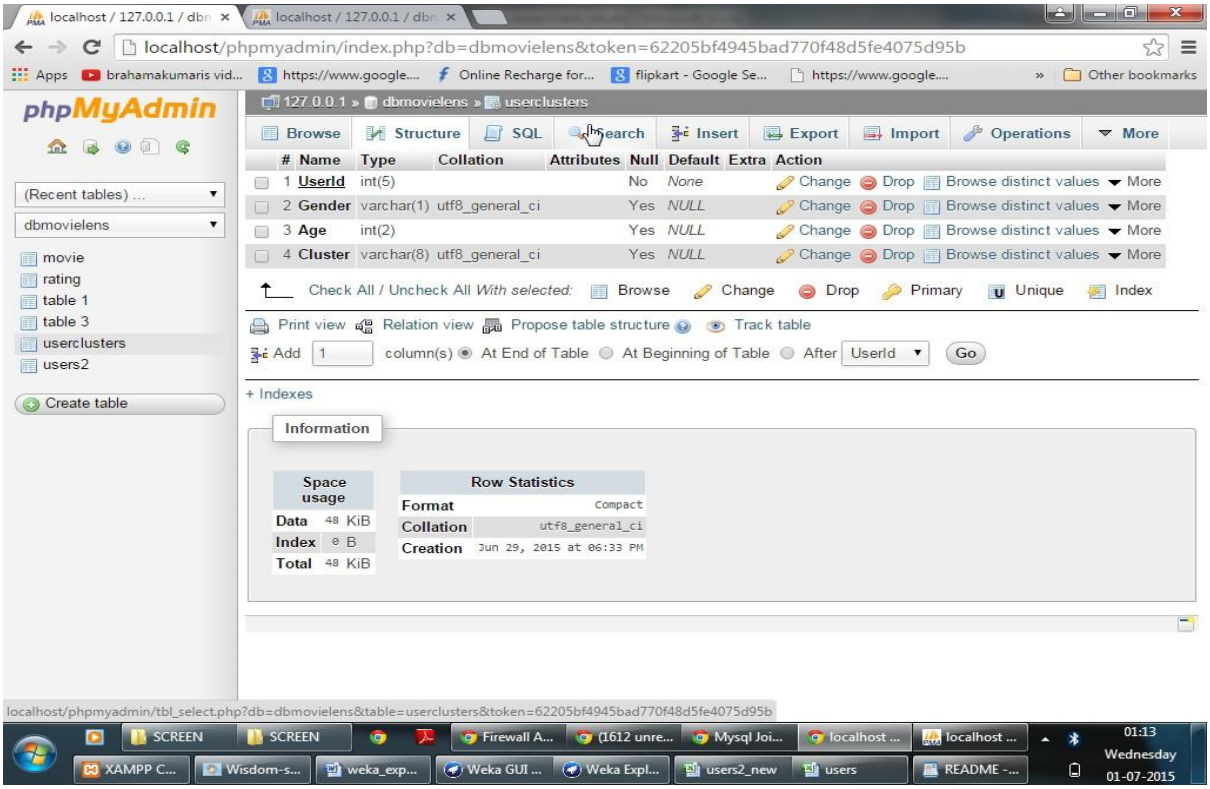

Figure 21: Schema of userclusters data table

Rating table is having two foreign keys i.e. UserId which refers to UserId of usersclusters table and MovieId which refers to MovieId of movie table

QUERY1: To incorporate these two foreign keys in rating table.
alter table rating add constraint fk_MovieId
foreign key (MovieId) references Movie(MovieId);

QUERY2: To find out the user rating based movie recommendations.

SQL result

Host: 127.0.0.1
Database: dbmovielens
Generation Time: Jun 30, 2015 at 04:10 PM
Generated by: phpMyAdmin 3.5.2.2 / MySQL 5.5.27
SQL query: SELECT rating.Movieid, movie.title, rating.rating, COUNT(rating.UserId) AS count FROM rating JOIN movie ON rating.Movieid = movie.MovieId WHERE rating.rating = '5' GROUP BY rating.Movieid ORDER BY count DESC LIMIT 0 , 30 ;
Rows: 30

Table 4: Query results showing count of users who rated ‘5’ to a particular movie

Movieid	Title	Rating	Count
318	Shawshank Redemption, The (1994)	5.0	91 [->]
296	Pulp Fiction (1994)	5.0	70 [->]
527	Schindler's List (1993)	5.0	69 [->]
593	Silence of the Lambs, The (1991)	5.0	55 [->]
110	Braveheart (1995)	5.0	54 [->]
50	Usual Suspects, The (1995)	5.0	54 [->]
260	Star Wars: Episode IV - A New Hope (1977)	5.0	53 [->]
858	Godfather, The (1972)	5.0	52 [->]
1198	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	5.0	47 [->]
608	Fargo (1996)	5.0	44 [->]
356	Forrest Gump (1994)	5.0	42 [->]
1196	Star Wars: Episode V - The Empire Strikes Back (1980)	5.0	37 [->]
1197	Princess Bride, The (1987)	5.0	34 [->]
1136	Monty Python and the Holy Grail (1975)	5.0	33 [->]
589	Terminator 2: Judgment Day (1991)	5.0	31 [->]
150	Apollo 13 (1995)	5.0	30 [->]
590	Dances with Wolves (1990)	5.0	28 [->]
47	Seven (a.k.a. Se7en) (1995)	5.0	28 [->]
750	Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)	5.0	26 [->]
1	Toy Story (1995)	5.0	25 [->]
541	Blade Runner (1982)	5.0	25 [->]
912	Casablanca (1942)	5.0	24 [->]
1193	One Flew Over the Cuckoo's Nest (1975)	5.0	23 [->]
1097	E.T. the Extra-Terrestrial (1982)	5.0	22 [->]
111	Taxi Driver (1976)	5.0	21 [->]
457	Fugitive, The (1993)	5.0	21 [->]
380	True Lies (1994)	5.0	21 [->]
480	Jurassic Park (1993)	5.0	20 [->]
364	Lion King, The (1994)	5.0	20 [->]

Movieid	Title	Rating	Count
923	Citizen Kane (1941)	5.0	19 [->]

QUERY3: To find out the user rating based movie recommendations after filtering the data on the basis of cluster information for the new user in the system whose rating record is not available.

The filtering on user ratings is done on the basis of user demographic information based appropriate cluster information for new user. It further helps in solving recommendation problem for the Cold Start Challenge of Recommender Systems where recommendations are based on ratings of similar users on the ground of UserId, Age and Gender.

SQL result

Host: 127.0.0.1
Database: dbmovielens
Generation Time: Jun 30, 2015 at 04:35 PM
Generated by: phpMyAdmin 3.5.2.2 / MySQL 5.5.27
SQL query:

```
SELECT rating.Movieid, rating.rating, movie.Title, COUNT( rating.UserId ) AS count
FROM rating
LEFT JOIN userclusters ON rating.UserId = userclusters.UserId
LEFT JOIN movie ON rating.MovieId = movie.MovieId
WHERE rating.Rating = '5' AND rating.UserId IN ( SELECT UserId FROM userclusters
WHERE Cluster = 'cluster4' )
GROUP BY rating.Movieid
ORDER BY count DESC LIMIT 0, 30;
```

Rows: 30

Table 5: Query results showing count of users who rated ‘5’ to a particular movie in a cluster to which a new user belongs (FIRST 17 RECORDS)

Movieid	rating	Title	Count
318	5.0	Shawshank Redemption, The (1994)	59 [->]
527	5.0	Schindler's List (1993)	47 [->]
296	5.0	Pulp Fiction (1994)	46 [->]
50	5.0	Usual Suspects, The (1995)	36 [->]
110	5.0	Braveheart (1995)	35 [->]
593	5.0	Silence of the Lambs, The (1991)	35 [->]
260	5.0	Star Wars: Episode IV - A New Hope (1977)	34 [->]

Movieid	rating	Title	Count
858	5.0	Godfather, The (1972)	32 [->]
608	5.0	Fargo (1996)	32 [->]
356	5.0	Forrest Gump (1994)	26 [->]
1196	5.0	Star Wars: Episode V - The Empire Strikes Back (1980)	24 [->]
1198	5.0	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	24 [->]
589	5.0	Terminator 2: Judgment Day (1991)	23 [->]
1197	5.0	Princess Bride, The (1987)	22 [->]
750	5.0	Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)	20 [->]
150	5.0	Apollo 13 (1995)	20 [->]
1136	5.0	Monty Python and the Holy Grail (1975)	20 [->]

5.10 Top 10 recommendation comparisons

There is a difference in recommendation preferences given for new user whose details are suppose

UserId	Age	Gender	Occupation	Zip-Code
501	M	25	2	55372

The new user belongs to cluster 4 as per k-means clustering algorithm based results using weka tool on user.csv dataset.

Table 6: Comparison between movie recommendations based on query results 2 and 3

Movieid from first query (on user rating info based)	Movieid from second query (user demographic cluster specific)
318	318
296	527
527	296
593	50
110	593
50	110
260	260

Movieid from first query (on user rating info based)	Movieid from second query (user demographic cluster specific)
858	608
1198	858
608	356
356	1196
1196	1198

Let recommendations are provided gender specific i.e. for male or female for the new user with specific details given below:

UserId	Age	Gender	Occupation	Zip-Code
502	30	F	4	55108

The new user belongs to cluster 3| as per k-means clustering algorithm based results using weka tool on user.csv dataset

SQL result

Host: 127.0.0.1
Database: dbmovielens
Generation Time: Jun 30, 2015 at 08:00 PM
Generated by: phpMyAdmin 3.5.2.2 / MySQL 5.5.27
SQL query:

```
SELECT rating.Movieid, rating.rating, COUNT( rating.UserId ) AS female_count
FROM rating
JOIN userclusters ON rating.UserId = userclusters.UserId
WHERE rating.Rating = '5' AND rating.UserId IN ( SELECT UserId FROM userclusters
WHERE ( Cluster = 'cluster3' AND Gender = 'F' ) ) GROUP BY rating.Movieid ORDER BY
female_count DESC LIMIT 0 , 30;
```

Rows: 30

Table 7: Top10 recommendations for new female visitor

Movieid	rating	female_count
318	5.0	10 [->]
110	5.0	9 [->]
356	5.0	8 [->]
527	5.0	8 [->]
593	5.0	8 [->]

Movieid	rating	female_count
47	5.0	7 [->]
296	5.0	7 [->]
1	5.0	7 [->]
590	5.0	7 [->]
858	5.0	7 [->]

Similarly we can find movie recommendations for males using SQL query:

SELECT rating.Movieid, rating.rating, COUNT(rating.UserId) AS male_count FROM rating

JOIN userclusters ON rating.UserId = userclusters.UserId

WHERE rating.Rating = '5' AND rating.UserId IN (SELECT UserId FROM userclusters WHERE (Cluster = 'cluster3' AND Gender = 'M')) GROUP BY rating.Movieid ORDER BY male_count DESC LIMIT 0 , 30;

Hence, the proposed system can better recommend movie choices if we consider user demographic information in addition to user ratings information in cold start problem which is a major challenge ahead to deal with for improving the efficiency of recommender systems and that too in less execution time due to less search space.

Further, other relevant attributes specific to problem domain can be considered for improving the quality and relevance of recommendations as per new user in the system.

The metric used here is count on users who rated movie with high rating i.e. ‘5’ and accordingly top ten recommendations are extracted from result which is ordered in descending order of count.

The experiments on MovieLens dataset performs well in generating recommendations following long-tail phenomenon i.e. it recommends even those items which are not found by users or not known to them [19].

Chapter 6

Conclusion and Future Scope

The proposed approach has resulted in reduction of overall execution time for generating recommendations and deals well with user cold-start problem. If n is the no. of users and m is the no. of items then the user-item rating matrix will be of the order of n*m and if we need to compute recommendations on user rating based recommendations then its complexity will be of the order of O(mn). The no. of ratings in total is quite high as compared to no. of users. The proposed methodology first of all analyzes the right combination of attributes for grouping dataset instances into clusters and then provides a sample of similar users on the basis of information contained in demographic attributes. These records constitutes neighborhood and as a consequence the search space for recommendation generation has reduced to great extent.

The aggregate function used for calculating Top-N recommendations is frequency count of users who rated a particular movie with highest score i.e. 5. The quality of recommendation generation also depends upon rating distribution and type of aggregate analysis. Rather than taking simple count based on highest rating frequency count another kind of aggregate functions can also be used say for example average rating for a particular movie.

Moreover, another set of demographic attributes can be exploited for finding clusters and hence recommendation accuracy can be improved. For this demographic attribute collection in user profiles can be increased for getting better recommendations. The information about user likes/dislikes, cultural background and other attributes can be used for personality diagnosis and community detection which is quite popular among researchers now a days for recommendation generation for an individual or for a group with same preferences.

In future, this methodology can be applied to other problem domains for generating recommendations and can also be used for new item cold-start problem. The dataset can be extended for analyzing the performance of recommender system in real-time. The only limitation in this approach is dependency upon user profile data and many users feels irritated and impatient in filling long forms. This limitation may further be solved using social tagging system in social networks.

REFERENCES

[1] www.ibm.com. [Online]. http://www.ibm.com/developerworks/library/os-recommender1/index.html

[2] Michael Hahsler, "recommenderlab: A framework for Developing and Testing Recommendation Algorithms ".

[3] Sunday O. Ojo, Seleman M. Ngwira and Keneilwe Zuva Tranos Zuva, "A Survey of Recommender Systems Techniques, Challenges," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 11, November 2012.

[4] TU Dortmund, Gerhard Friedrich Dietmar Jannach. (2013, August) Recommender Systems.

[5] Xiaoyuan Su and Taghi M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, August 2009.

[6] Ron Zacharski. A Programmer Guide to Data Mining : The Ancient Art of the Numerati.

[7] Laila Safoury and Akram Salah. (2013, August) Exploiting user demographic attributes for solving cold-start problem in recommender system.

[8] Amit Sharma. What are some of the most interesting research problems in Recommender Systems? [Online]. http://www.quora.com/What-are-some-of-the-most-interesting-research-problems-in-Recommender-Systems

[9] How to Evaluate Machine Learning Models: Classification Metrics. [Online]. http://blog.dato.com/how-to-evaluate-machine-learning-models-part-2a-classification-metrics

[10] Alan Said, Domonkos Tikk, and yue Shi, "Recommender Systems Evaluation: A 3D Benchmark," in *Workshop on Recommendation Utility Evluation*.

[11] www.grouplens.org. [Online]. http://grouplens.org/datasets/movielens/

[12] Javad Basiri, Azadeh Shakery, Behzad Moshiri, and Morteza Zi Hayat, "Alleviating the Cold-Start Problem of Recommender Systems Using a New Hybrid Approach," , 2015.

[13] Honey Jindal and Sandeep Kumar Singh, "A HYBRID RECOMMENDATION SYSTEM FOR COLD-START PROBLEM USING ONLINE COMMERCIAL DATASET," *International Journal of Computer Engineering and Applications*, vol. VII, no. I, July 2014.

[14] Zhang Zi-Ke, Chuang Liu, Yi-Cheng Zhang, and Tao Zhou. (2010, October) Solving the cold-start problem in recommender systems with social tags.

[15] Suchismit Mahapatra, Alwin Tareen, and Ying Yang. A Cold Start Recommendation System Using

Item Correlation and User Similarity.

[16] Vinod Krishnan, Pradeep Kumar Narayana, Mukesh Nathan, Richard T. Davies, and Joseph A. Konstan. Who Predicts Better? – Results from an Online Study Comparing Humans and an Online Recommender System.

[17] Shaghayegh Sahebi and William W. Cohen. Community-Based Recommendations: a Solution to the Cold Start Problem.

[18] www.mathworld.wolfram.com. [Online]. http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html

[19] Zdravko Markov and Ingrid Russell. [Online]. http://www.cs.ccsu.edu/~markov/weka-tutorial.pdf

[20] Rushdi Shams. Weka Tutorial 02 : ARFF101(Data Preprocessing).

[21] Mark Polczynski. Weka Tutoral - Clustering.

[22] Mark Polczynski. Weka Tutorial 2 - Data Mining Concepts.

[23] Ian H. Witten. More Data Mining with Weka(3.5 : Representing Clusters).

[24] Ian H. Witten. More Data Mining with Weka(3.6 : Evaluating Classes).

[25] Mark Polczynski. Weka Tutorial3 - Classification using Decision Trees.

[26] Shalini Batra, "Recommender Sytem - An Introduction,".

LIST OF PUBLICATIONS

- Shano Solanki and Shalini Batra, "Recommender System using Collaborative Filtering and Demographic Characteristics of Users", 2015 International Journal of Recent and Innovative Trends in Computing and Communication (IJRITCC) ISSN 2321-8169 [Accepted].
- Shano Solanki and Shalini Batra, "Hybrid Recommender System for solving User Cold-Start Problem using Collaborative Filtering and Demographic Information of Users", International Conference on Advanced Computing and Communication Technologies (ICACCT 2015) [Communicated].