A HYBRID UNIFIED FRAMEWORK TO RECOMMENDER SYSTEM IN E-COMMERCE USING MULTI LAYER ARCHITECTURE THROUGH CLUSTERING

[1] *RUSHIRAJ R. BORISAGAR,* [2] *DR. BHAVIN S. SEDANI,* [3]*DR. VIPUL M. VEKARIYA*

[1] Research Scholar, Department of Computer Engineering, C. U. Shah University, Gujarat
[2] Professor and Head of Electronics & Communication Department, V.V.P. Engineering College, Gujarat
[3]Asso.Prof. and Head of Computer Department, Noble Engineering College, Gujarat

r_borisagar@yahoo.co.in, bhavin_s_sedani@yahoo.com, vekariya.vipul@gmail.com

ABSTRACT : *The exponential growth of the Internet and the increasing popularity of e-commerce services provide users with a vast amount of information and choices. This information overload problem has been addressed by recommender systems. Recommendation systems have arisen to provide convenient suggestions to the users. These systems can be used for different purposes in several domains from offering papers to researchers to helping consumers in e-commerce. There are recommendation systems in different domains such as films, television programs, video, music, books, news, images, and web pages. It needs a new approach that can deliver good results in dynamic problem spaces. Objective of this research paper is to integrate the existing recommender approaches in one framework and combine them in a flexible and adaptable manner, architecture called MLARS*

.

KEY WORDS: Recommender System, Collaborative, Content Based, Hybrid, Clustering.

1. Introduction

Different areas of research have addressed the information overload problem and given rise to new approaches that support the user in finding relevant information and making better choices. Nowadays, advanced retrieval services, recommender and filtering systems are available to the users and their widespread and frequent use proves their utility. However, each of the filtering techniques is designed for a septic setting of the problem space. But what happens when the problem domain is dynamic and changes over time? Then, we are talking about dynamic problem spaces where some characteristics of the domain, such as the amount of available information or the current context of the system, are changing over time. To compensate for this change, we would either have to switch to another filtering method or adjust its contribution when we are combining several methods.

Types of recommender system

Recommender systems are divided according to their approach to rating estimation. The Recommender systems are classified into the following categories[5]

•**Content-based recommendations**: Based on past history[6]
•**Collaborative recommendations**: Based on similar test and preference[7].
•**Hybrid approaches**: combines more than one method[8]. (Collaborative and content-based)

One single filtering technique or the static combination of two techniques does not meet the challenges of today's dynamic world. CF, being the prevalent personal filtering technique, needs a considerable amount of interaction between the users and the employed system before new items of interest can be suggested. Furthermore, it is computationally expensive and therefore not suitable for real-time interaction. Thus it needs a different approach, until we have sufficient information about the users, such as CBF which is independent from the user. The systems should be able to adjust itself depending on the amount of information available about a particular user.

Framework allows making use of the combined strength of the different filtering methods. Moreover, the framework should be portable to distributed scenarios, such as personal recommender systems, in which case it allows only partial set of the whole information space. Such a personal recommender system should provide its users with recommendations wherever they are and whenever they request it. In this scenario, each user is carrying their own

personal recommender system and the system should be updatable without the assumption of a centralized infrastructure. In addition, such a recommender system must run fast and efficiently on mobile devices which have limited memory and operate with slow processors. So the main problems in traditional filtering techniques are

- The combination of the two techniques is static
- Does not adjust itself when the state of the problem domain changes[5]
- Hybrid methods can only combine two techniques in one single framework
- The hybrid methods consume large amounts of memory and require high processing power & time consuming
- First rater Problem[2]
- Sparsity Problem[1]

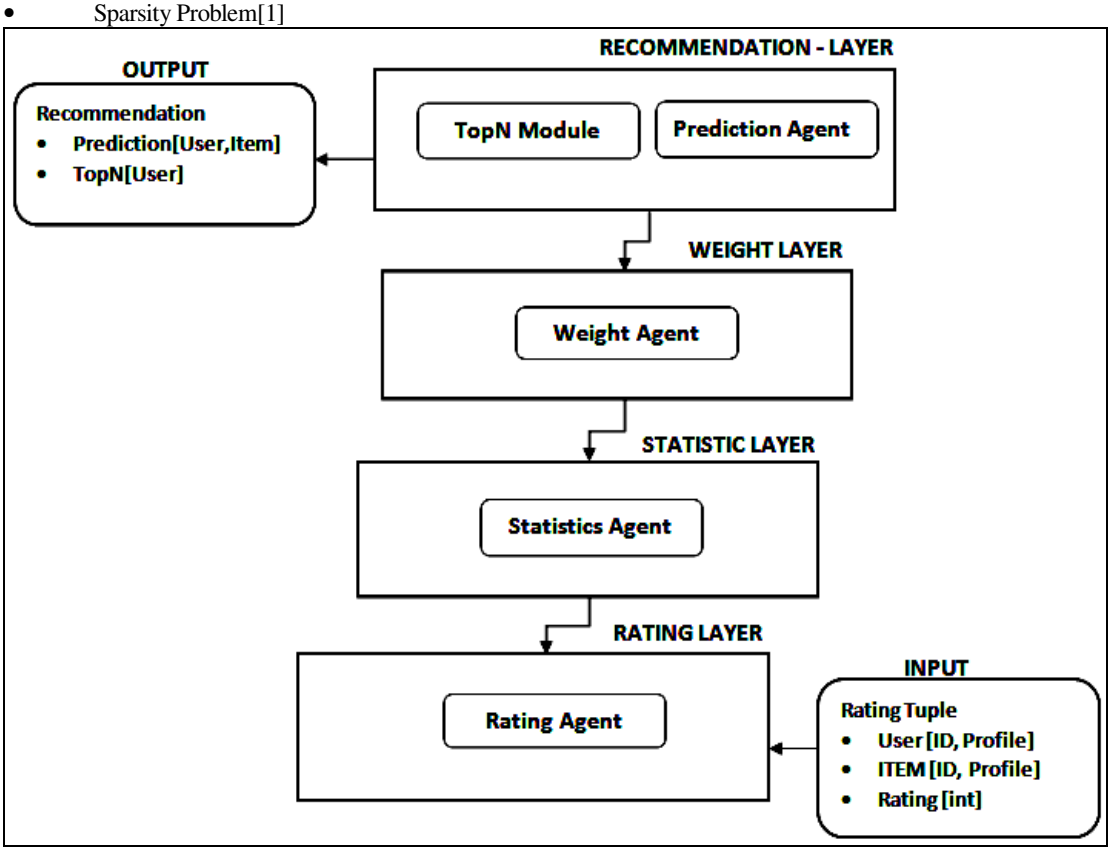

Figure: 1- Form Modules to MultiLayers

2. MLARS (Multi Layered Architecture for Recommender Systems)

Therefore, we propose a novel architecture, MLARS, which stands for Multi Layered-Architecture for Recommender Systems, that allows combining several filtering approaches in a transparent and flexible way in one single framework. With MLARS it can from its strength. It has been inspired by the Internet that uses a layered architecture to deal with the complexity of data transmission over heterogeneous media between different entities. In this case the layered architecture allows a maximum of flexibility to accomplish the task of data transmission between entities despite varying physical media and different end-systems. Our goal is to build a model that provides a similar flexibility as the Internet to accomplish its task. In this case the task is to provide recommendations for users and the complexity stems from integrating several filtering techniques in one architecture. Adopting the idea of the layered architecture helps us to break down the complexity of generating recommendations. Similar to the TCP/IP model [10], MLARS consists of several layers where each layer represents a functional division of the tasks required to generate recommendations. Every layer is charged with performing a certain subset of the total functionality required to implement a recommender system.

The abstraction in a layered-architecture allows to integrate several filtering algorithms within one framework. Furthermore, MLARS is not restricted to a combination of two recommendation techniques and can control the contribution of any method to the granularity of each prediction. In a layered architecture, each layer represents a functional division of the task by realizing a subset of services that in-turn can be used by other layers. However, in a layered architecture the interaction between layers is restricted.

- Each layer only relies on services provided from the layers below.
- Each layer only offers its services to the layer above.

Note that in our interaction scheme, each layer can also access layers at lower levels, whereas in the layered architecture of the Internet, each layer can access only the layer immediately below it.

Furthermore, layers define interfaces for interaction between them. It is important to point out that the interfaces of a layer are fixed, but its implementation is not. Therefore, we guarantee that a layer can be exchanged as long as the interfaces are implemented or in other words the same services are provided. An upper layer is relying on the services provided by the lower layers and uses its services as a starting point to implement its own service.

PROPOSED MLARS ARCHITECTURE

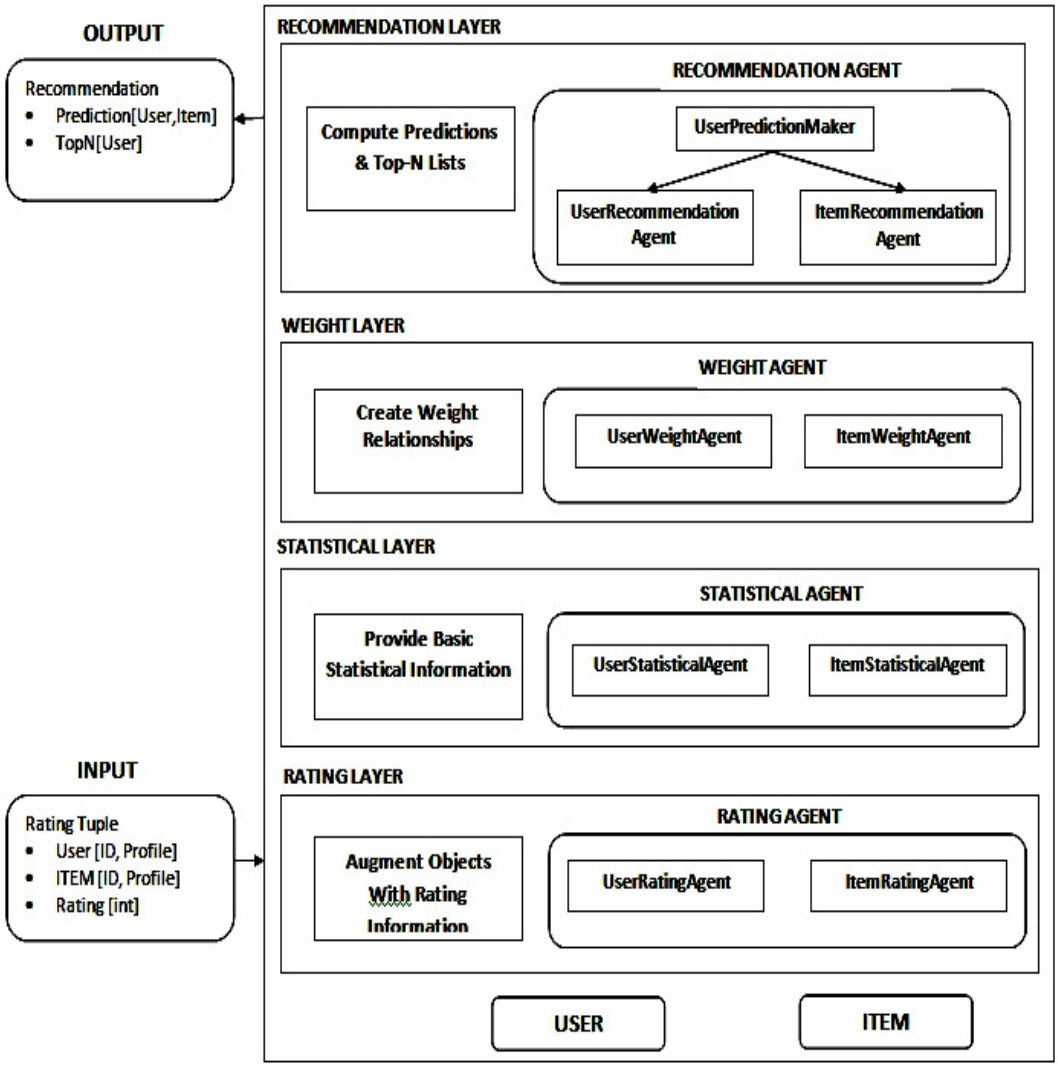

Figure: 2 – MLARS Architecture

MLARS consists of the four layers. Each layer is charged with performing a certain subset of the entire functionalities required to implement a recommender system. They work together to allow the recommender system as a whole to operate. Each layer provides its service by creating its own independent object. These newly created objects are then passed to the upper layer to help it accomplish its task.

Rating-Layer

- **Service**. The Rating-Layer aggregates the rating information that is provided by the input data and creates a UserRatingObject and an ItemRatingObject. This layer attaches the rating information to the corresponding user or item objects and makes it thereby accessible in a convenient way for higher layers.
- **Output Interface.** The interface to access the provided information for the layer above is encapsulated in a UserRatingObject and an ItemRatingObject.
- **Input Interface.** The input to the Rating-Layer are rating tuples.

Algorithm updateUserRatingAgent(RatingTuple)
Input: RatingTuple(User user, Item item, Rating rating)
Output: Updated UserRatingAgent

//Returns the corresponding UserRatingAgent
UserRatingAgent ura ← findRatingAgent(user)
// update number of ratings
ura.ratings ← ura.ratings + 1
// update rating vector
ura.ratingVector.add(item, rating)
return ura

Statistics-Layer

- **Service.** The responsibility of the Statistics-Layer is to provide basic statistical information that is based on the ratings given and received by the users and items, respectively. Statistical information can be, for instance the average, the standard deviation or the vector length of the provided ratings.
- **Output Interface.** The statistical information is encapsulated by UserStatisticsObjects and ItemStatisticsObjects and thereby accessible for higher layers.
- **Input Interface.** The input data to the Statistics-Layer are the two objects created by the Rating-Layer, namely UserRatingObjects and ItemRatingObjects.

Algorithm updateUserStatisticsAgent (UserRatingAgent,ItemRatingAgent)
Input: UserRatingAgent ura, ItemRatingAgent ira
Output: Updated UserStatisticsAgent

// iterate over all UserStatisticsAgent
for each (UserStatisticsAgent usa) do
{
if (usa = ura) then
{
// update average
usa.avgSum ← usa.avgSum + ura.Rating(ira)
usa.avg ← usa.avgSum / ura.NumberOfRatings
// update vector length
usa.partialVLen ← usa.partialVLen + ura.Rating(ira).square
usa.vLen ← usa.partialVLen.root
// update standard deviation
usa.std ← update(usa.RatingVector, usa.Avg, usa.NumberOfRatings)
}else
{ // update co-ratings: if usa has rated and ura != this
if (usa.Rated(ira)) then
{
usa.CoRatingVector(ura) ← usa.CoRatingVector(ura) + 1
}
} return usa
}

Weight-Layer[14]

- **Service.** The task of the Weight-Layer is to define relationships between users and items. Weights are numerical values and can represent a wide variety of relationships such as correlation in taste or overlap in content information.
- **Output Interface.** The information about the weight-relationship among user and item objects is encapsulated by UserWeightObjects and ItemWeightObjects, respectively.

- **Input Interface.** The weight-objects are based on the information provided by UserStatisticsObjects and ItemStatisticsObjects.

Algorithm updateUserWeightAgent(UserStatisticsAgent usa, ItemStatisticsAgent isa)
Input: UserStatisticsAgent usa, ItemStatisticsAgent isa
Output: Updated UserWeightAgent

```
// iterate over all UserWeightAgent
for each (UserWeightAgent uwa) do
{
// adjust weights only when uwa rated item
if (uwa.Rated(isa))    then
{
// check weight to each neighbour
for each (UserWeightAgent neighbour : uwa.Neighbours) do
{
// weight has to be adjusted if uwa rated item, then dot-product changes
if (neighbour.Rated(isa))
{
// compute partial dot-product
uwa.partialDotProduct ← uwa.partialDotProduct + (uwa.Rating(isa)*neighbor,R
}
// compute cosine weight, vector length already updated by lower-layer
uwo.weight ← uwo.partialDotProduct / (uwa.VLen * neighbour.VLen)
// update weight relationship
uwa.WeightVector.set(neighbour, weight)
}
// return updated UserWeightAgent
return uwa
}
}
```

Recommendation-Layer

- **Service.** Once the weights are established and the rating information is made accessible, the task of the Recommendation-Layer has become easy. Based on the recommendation strategies, it can do both infer ratings for items or create a topN-list for a target user.
- **Output Interface.** The service of generating recommendations is done by the UserPredictionMaker which implements the general output interface for recommender systems(see Figure 4).
- **InputInterface.** To provide recommendations the Recommendation-Layer needs either UserWeightObjects or ItemWeightObjects, or in a hybrid approach both.

Algorithm computePrediction(user,item)
Input: User user, Item item
Output: Prediction for item

```
// Returns the corresponding UserRecommendationAgent
UserRecommendationAgent ura ← findUserRecommendationAgent(user)
// Returns the corresponding ItemRecommendationAgent
ItemRecommendationAgent ira ← findItemRecommendationAgent(item)
// determine lamda depending on current situation
lambda ← lambdaStrategy(ura,ira)
// compute a prediction
prediction ← lamda * ura.prediction(item) + (1 – lambda) * ira.prediction(item)
// returns the final prediction for item
return prediction;
```

Algorithm computeTopN(user)
Input: User user, Item item
Output: Prediction for item

```
// Returns the corresponding UserRecommendationAgent
UserRecommendationAgent ura ← findUserRecommendationAgent(user)
// Returns all ItemRecommendationAgent
Collection<ItemRecommendationAgent> ira ← getAllItemRecommendationAgent( )
// compute topN-list
topN ← topN-Algorithm(ura,ira)
// returns the final topN-list
return topN;
```

3. Domain Description

We demonstrate the working of our approach in the domain of movie recommendation. The dataset contains rating data provided by each user for various movies. User ratings range from zero to five stars. Zero stars indicate extreme dislike for a movie and five stars indicate high praise. These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. Each user has at least 20 ratings.

We represent the content information of every movie as a set of slots (features). Each slot is represented simply as a bag of words. The slots we use for the Each Movie dataset are: movie title, director, cast, genre, plot summary, plot keywords, user comments, external reviews, newsgroup reviews, and awards.

4. Methodology

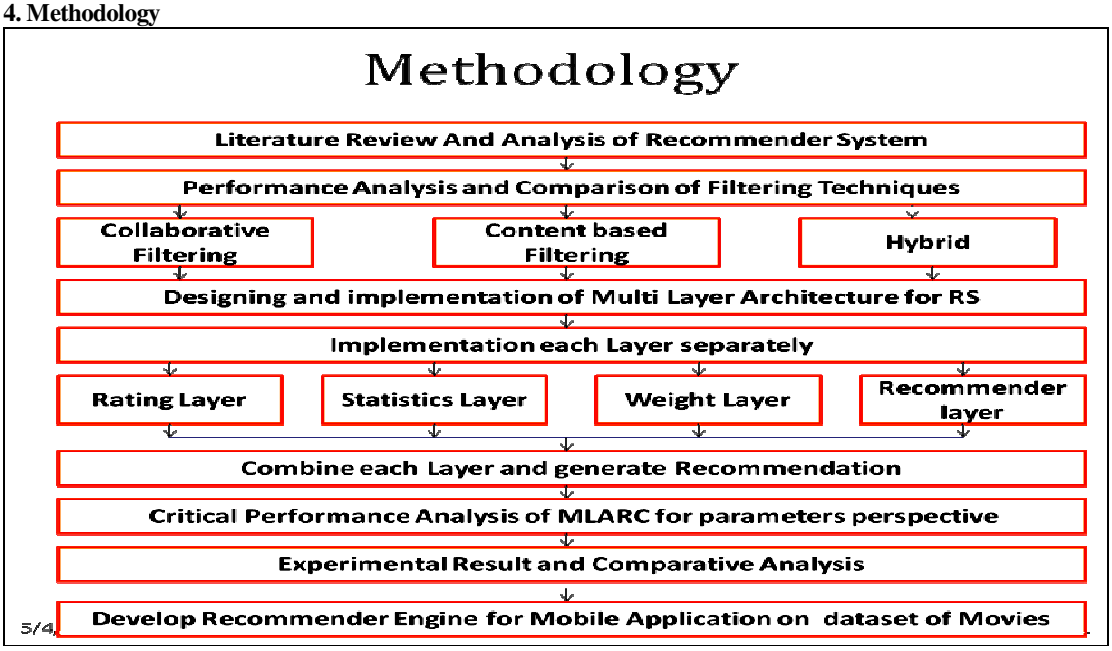

Figure: 3- Methodology for MLARS

5. Metrics

The metrics for evaluating the accuracy of a prediction algorithm can be divided into two main categories: statistical accuracy metrics and decision-support metrics[15]. Statistical accuracy metrics evaluate the accuracy of a predictor by comparing predicted values with user-provided values. To measure statistical accuracy we use the mean absolute error (MAE) metric—defined as the average absolute difference between predicted ratings and actual ratings. In our experiments we computed the MAE on the test set for each user, and then averaged over the set of test users. In statistical analysis, another metric is the F1 score (also F-score or F-measure). It is a measure of a test's accuracy. F-measure considers both the precision p and the recall r of the test to calculate the score: p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned. The F1 score or F-measure can be interpreted as a weighted average of the precision and recall, where an F-measure reaches its best value at 1 and worst score at 0. Decision-support accuracy measures how well predictions help users select high-quality items. We use Receiver Operating Characteristic (ROC) [4] sensitivity to measure decision-support accuracy. A predictor can be treated as a filter, where predicting a high rating for an item is equivalent to accepting the item, and predicting a low rating is equivalent to rejecting the item. The ROC sensitivity is given by the area under the ROC curve — a curve that plots sensitivity versus 1-specificityfor a predictor. Sensitivity is defined as the probability that a good item is accepted by the

filter; and specificity is defined as the probability that a bad item is rejected by the filter. We consider an item good if the user gave it a rating of 4 or above, otherwise we consider the item bad.

6. CONCLUSION AND FUTURE WORK

MLARS architecture is powerful enough to integrate several recommendation approaches in a transparent and flexible way in one single framework. This allows to use the combined strengths of the various recommendation algorithms and to overcome the limitations of today's hybrid approaches.

We will develop real time Android based mobile application using framework - Multi Layer Architecture of Recommender on data set of Movies and perform final analysis of results of all proposed techniques with traditional filtering techniques.

7. REFERENCES:

1. Zeinab Abbassi, Sihem Amer-Yahia, Laks V.S. Lakshmanan, Sergei Vassilvitskii, Cong Yu, "Getting Recommender Systems to Think Outside the Box", In the Proceedings of the third ACM conference on Recommender systems, pp. 285-288, 2009.

2. Wang, Y., Stash, N., Aroyo, L., Hollink, L. and Schreiber, G., "Using Semantic Relations for Content-based Recommender Systems in Cultural Heritage", Workshop on Ontology Patterns (WOP) at International Semantic Web Conference (ISWC), October, 2009.

3. Billy Yapriady, Alexandra L. Uitdenbogerd, "Combining Demographic Data with Collaborative Filtering for Automatic Music Recommendation", Knowledge-Based Intelligent Information and Engineering Systems, pp. 201-207, 2005.

4. Blanco-Fernandez, Y., Pazos-arias, J., Gil-Solla, A., Ramos-Cabrer, M., Lopez Nores, M., "Providing Entertainment by Content-based Filtering and Semantic Reasoning in Intelligent Recommender Systems", IEEE Transactions on Consumer Electronics, 2008.

5. Master Thesis, Mahir Yildirim. "Towords a unified framework for recommender system", Swiss Federal Institute of Technology,Zurich, 2007.

6. Yiwen Wang , Natalia Stash , Lora Aroyo , Laura Hollink , Guus Schreiber, "Semantic Relations in Content-based Recommender Systems", Proc. 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, CA, pages 209-210. ACM, 2009.

7. Chikhaoui B,Chiazzaro M, Shengrui Wang," An Improved Hybrid Recommender System by Combining Predictions", Advanced Information Networking and Applications (WAINA), IEEE Workshops of International,2011.

8.Yu Liangxing, Dong Aihua," Hybrid Product Recommender System for Apparel Retailing Customers", Information Engineering (ICIE), WASE International Conference(Volume:1), 2010.

9. Ragad A.H.M., Mashat A.F.S., Khedra A.M," HRSPCA: Hybrid recommender system for predicting college admission", Intelligent Systems Design and Applications (ISDA), 12th International Conference, 2012

10. Ghazanfar M.A., Prugel-Bennett A.," A Scalable, Accurate Hybrid Recommender System", Knowledge Discovery and Data Mining, WKDD '10. Third International Conference,2010

11. Verma, S.K., Mittal, N., Agarwal, B." Hybrid recommender system based on fuzzy clustering and collaborative filtering", Computer and Communication Technology (ICCCT), 4th International Conference,2013

12. DongtingS.,Cong L.,ZhigangL."A Content enhanced approach for cold-start problem in collaborative filtering", 2011.

13. Putu Wira uana,Sesaltina Jannet D.R.M,I Ketut Gede Darma Putra,"Combination of K-Nearest Neighbor and K-Means based on Term Re-weighting for Classify Indonesian News", International Journal of Computer Applications (0975 – 8887) Volume 50 – No.11, July 2012.

14. Urszula Kużelewska,"Clustering Algorithms in Hybrid Recommender System on MovieLens Data". Studies in Logic, Grammar and Rhetoric ,2014,pg 125-139, 2014.

15. Hao Ma,Dengyong Zhou,Chao Liu,Michael R. Lyu,Irwin King," Recommender Systems with Social Regularization" January 2011.