

Reducing the Cold-Start Problem by Explicit Information with Mathematical Set Theory in Recommendation Systems

Haider Khalid¹, Shengli Wu²

¹Jiangsu University, School of Computer Science and Communication Engineering,
Xuefu Road, Zhenjiang 212013, China
m.haiderkhalid@hotmail.com

²Jiangsu University, School of Computer Science and Communication Engineering,
Xuefu Road, Zhenjiang 212013, China
1000004032@ujs.edu.cn

Abstract: *The explosive growth of enormous information makes our world as a global village. Recommendation systems are widely used in many different kinds of commercial web sites. A key challenge is how to provide recommendations when historical data for a user is missing and mining relationship between user and recommender system, a problem generally known as cold-start recommendation. This issue is a well-known problem for recommender systems and much research has been done to find the best way to overcome this. The proposed idea is querying user through an initial interview to get explicitly information, this process proposed as new user preferences to make user profile. In this paper, we present mathematical set theory to solve cold-start recommendation problem. Experiment will perform on Movie-lens dataset and user preferences can be formed into mathematical set. The combinations of multiple preferences represent as subsets and the largest subset will be the first choice as input for the recommender system. User Item matrix strategy will be used to get predicted rating for cold users. Experimental results on mathematical set approach for recommendation on the Movie-lens dataset demonstrate that the proposed approach significantly outperforms existing methods for cold-start recommendation.*

Keywords: Cold-Start, Recommender System, Mathematical Set Theory, Explicit Information, Content-Based Filtering.

1. Introduction

Since about 1995, the recommendation systems have been deployed across many domains. Two of the most earliest and representative recommender systems are Ringo (publicly available in 1994) and Group-Lens (available in 1996). Ringo and its success on the first large scale were on the music recommendation systems [1]-[2]. On the other hand, Group-Lens was an automated collaborative filtering system for the Unset News and also very successful. The trial of the Group-Lens recommender system showed that the collaborative filtering method could be effective on large scale of data [2]. Meanwhile the Group-Lens project was adapted to produce Movie-Lens, a large number of collections available for movies recommendation systems. Large interest in recommender systems was soon fostered by the increasing public demand for systems that helped deal with the problem of information overload. Since then, most of the academic and commercial interest has been shown in recommender systems for many different domains. Although much of their research was not published but now Amazon.com is one of the most well-known implementer of this technology. Amazon.com use collaborative filtering systems to recommend products that a user might like to purchase. Other companies that use recommender systems include netflix.com for videos, TiVo for digital television, Barnes and Noble for books. Many music recommendation systems are also available today, such as Pandora.com (which maintains a staff of music analysts who

tag songs as they enter the system) and Last.fm3. These two systems are considering the best music recommenders currently available to the public [3].

By composing the recommendations to the user for particular

items is usually done by one of the two strategies (Pandora.com and Last.fm3). Either we can see the content of the item itself or by matching it against the use profile by content-based filtering or predicting a rating based on other users rating on the items. These strategies are having the same kind of challenges relating to the cold-start problem.

To get to know about the new user in the recommendation system, we need to get knowledge and study about different methods to collect the data about the user acts and the way he is using the application or ask the peer users to add some additional information [4]. We also want to develop an application and design the structure to describe the data ourselves. That application is able to build knowledge about the users and items in an efficient way. Many datasets exist to be used for testing recommender systems, but those datasets contain already historical data, typical user ratings on items like books, music or movies. To develop a real world application, we can be able to collect the both implicit and explicit ratings, look at the content of the items ourselves and therefore be more flexible in design of the recommendation system. The best domain is that we don't have any knowledge about the user but can be able to extract some information and after analysis can turn some resourceful recommendations. Thinking of a domain where people can make choices on a regular basis according to their interest and there are a lot of options out there that takes some effort for a person to explore. Deploying and implementing an application with movies information along with a recommendation system, we can help people to save their time. It is also difficult to know that what kind of movies for an unknown person can like, so this phenomenon can gives us a good domain for studying the cold-start problem.

2. Related Work

Since the recommendations are considered to target for an individual users and keeping the data for all users in the system, it is impossible to know about how to make good recommendations for a new user just entered to the system and system has no knowledge about this user. In the starting by giving irrelevant or bad recommendations, the user might be scared and system might be possible to lose the user trust. He could get disappointed by the mistakes of the recommender system, so he will possibly leave and quit using the application before enough knowledge about the user is built up for the system to get accurate and enrich recommendations [5]. A crucial feature of most recommender systems is therefore to extract enough knowledge about the new user to be accurate enough. Many different techniques and methods to solve this cold-start problem have been tried out, and we will describe some of the most common ways to do this.

2.1 Content-Based Filtering

A content-based filtering method is one of the most frequent used methods that analyzes the items itself and match them against a user profile [6]. The representation of an item to a user is important in content-based method because the recommendation technique probably based on the representation. In a structured technique, the normal attributes like tags or categories about a user can be easily stored, so that is easy to compute from them. The more complicated is unrestricted text, by counting words and represent their importance; an unrestricted text can be transformed into structured data. This process is often done by representing a weight for each term TFIDF (term frequency time's inverse document frequency). Before done this process use the common techniques like stemming, which means the representation of different versions of single word as one term. For example, "com" can be representing the terms as "computation" and "computers". Since an item is similar with user interest. The recommender system has a technique to compute a score for a user on the items that describes the two types of user profiles [6]. Firstly, can be the user's preferences are stored. Preferences like this could be collected by having the user fill out a profile with explicitly that we will also discuss in this paper as proposed approach. Secondly, the history of the user can be interacting with the recommender system. This information might be history like which items he has looked at before, ratings, or queries. This feedback can be collected either implicitly or explicitly.

2.2 Collaborative Filtering

Collaborative filtering method is a recommender approach based on a collection of the user's data like ratings, behaviors, and preferences, as well as analyzing these data and recommend items to a user. The task can be described as "To predict the utility of items to a particular user (the active user) based on a database of user votes from a sample or population of other users (the user database)" [7]. Collaborative filtering can be in different forms; memory based or model-based methods. In memory-based algorithms, we usually predict the votes for an active user based on some partial information regarding to that active user and a set of weights calculated from the user database. The user database contains a set of votes $V_{u,i}$ meaning the vote for user u on item i . If I_u is the set of items on which user u has voted, then the mean vote for

user u is as follows:

$$\bar{v}_u = \frac{1}{|I_u|} \sum_{i \in I_u} v_{u,i} \quad (1)$$

2.2.1 Memory-based Algorithm

We predict the votes of the active user based on partial information regarding the active user and from the database we can get a set of weights. $P_{u,i}$ is a weight of sum of the votes of the users:

$$P_{u,i} = \bar{v}_u + k \sum_{u' \in U} w(u, u') (v_{u',i} - \bar{v}_{u'}) \quad (2)$$

Where u is the current user, U is the set of users in the database with nonzero weights. The weights, $w(u, u')$ can describe the correlation or similarity.

Table 1: User x Item Rating Matrix

	Item A	Item B	Item C	Item D
User A		1		5
User B	1	?	3	?
User C	2		4	5

The goal for the "Table 1" is to predict the missing votes for the active user B, between each user in the database and the active user k is for normalization. In this paper, we will cover the correlation coefficient for selecting neighbors. The correlation between two users' u and u' is defined as follows:

$$w(u, u') = \frac{\sum_i (v_{u,i} - \bar{v}_u)(v_{u',i} - \bar{v}_{u'})}{\sqrt{\sum_i (v_{u,i} - \bar{v}_u)^2 \sum_i (v_{u',i} - \bar{v}_{u'})^2}} \quad (3)$$

Where i is for each item for which both user u and u' have recorded votes.

2.2.2 Model-based Algorithm

This algorithm is using a probabilistic approach where the collaborative filtering task can be viewed as calculating the expected value of a vote from the knowledge about the user. For the active user, we want to predict the votes for unobserved items. The probability expression for an item i and user u with the rated items for user u is I_u is as follows (this assumes that the votes are integers in range from 0 to m), $P_{u,i} = E(v_{u,i})$

$$E(v_{u,i}) = \sum_{j=0}^m \Pr(v_{u,i} = j | v_{u,k}, k \in I_u) i \quad (4)$$

The probability, Pr in (4) is the probability that the given user will rate a value of the input item, given the previously rated items. Bayesian networks and clustering models is two of the probabilistic models that can be used for model-based collaborative filtering. Collaborative filtering helps with predicting ratings for an active user based on previous votes in a database of votes. The input data is very often a sparse matrix of votes for different items. The example in Table 1 is a small example showing ratings for some users, and where the data is sparse.

2.3 Demographic Technique

In this technique user can either enter some information explicitly, or connect with some social media and retrieve the information. This can help the user for grouping into similar groups. Depending on what type of information needed to the system, some groups of people might share the same interests, so grouping people by attributes like age, sex, nationality, occupation, city, education, income, marital status, might help for recommendations. By asking to the user to enter all required information into the system is a little time consuming for the user, and might make him tired. By considering at the domain of the recommender system you want only to collect the data that will be most relevant and valuable to give recommendations. Today's more and more applications adapting the technique and allow users to sign up with their social network profile, and this technique could help them by retrieving their data without entering it manually.

While demographic data can help in some specific recommender systems, it is also less general than other techniques so it depends on domain information and only applies to certain domains as [8] using another approach to the cold-start problem.

2.4 Tags

Collaborative filtering method typically uses a matrix technique User \times Item, which will be difficult to fill out for a new user in to the system. If the we can grouped the items into tags, a matrix will be change from User \times Tag, then the tags will be worthy, and you know the users opinion about every tag then you will be able to transform that information towards multiple items. This kind of approach was introduced by [9] and they tested this technique on a dataset of bookmarks, a site containing bookmarks tagged by the users. They achieved even better results than comparing with regular collaborative filtering on the items when measured with recall.

Their approach to tackle the problem was first to build a model for the candidate tags for a user, using the collaborative filtering method. Using that model of the tags, they generate the top-N recommendations using Naive Bayes.

H. Kim et al. [9] mentioned the issue of noisy tags makes the performance of recommendation worse with their method. This issue is also be a problem for some real world applications in where the users create the own tags with their names or irrelevant with the real word and that increase the number of tags in the system and increase more data to handle them.

2.5 Hybrid System

Hybrid recommender system is used to tackle the certain limitations of content-based- or collaborative filtering methods. Collaborative filtering methods have a relatively big drawback about cold-start problem since the user is required to rate some items to get the recommendations from the system. A real-life problem domain is much more complex than a recommender system for movies [10]. In real business case example recommender system will have significant amounts of parameters, where a hybrid technique can help learning a more complex model of the user, hence giving more accurate and enrich recommendations. Hybrid solutions are a mixture of the methods described earlier like collaborative and content-based filtering. Multiple hybrid solutions can be created by combining the different methods.

2.6 User Select Some Trusted Users'

Another approach was also considered that has been used is making the user select one or multiple users that he trusts and relevant to his interest, and then get the recommendations based on that users information [11]. This technique can gives the explicit information to recommender system as an initial input about the new user that can be valuable for recommending items. But the problem is the trusted user might change their interest after some time and the actual user get the wrong recommendations.

3. Architecture Model

The whole architectural model for our experiment approach and experiment performance on movie-lens dataset for movies recommendation will describe in this section. The model is based on the new user just enter to the system and system has no knowledge about this new user to tackle this cold-start problem that we already discussed and describe previous section. As mentioned earlier, we found it necessary to develop our application to promote flexibility to what data we can use for the recommendations.

3.1 Proposed Approach

When a new user just enters into the system, the user knowledge about the system probably none and also the system does not know any information about the user, the "Figure 1" describes the interaction with a new user and a recommender system to know about each other's to recommend and get recommendations.

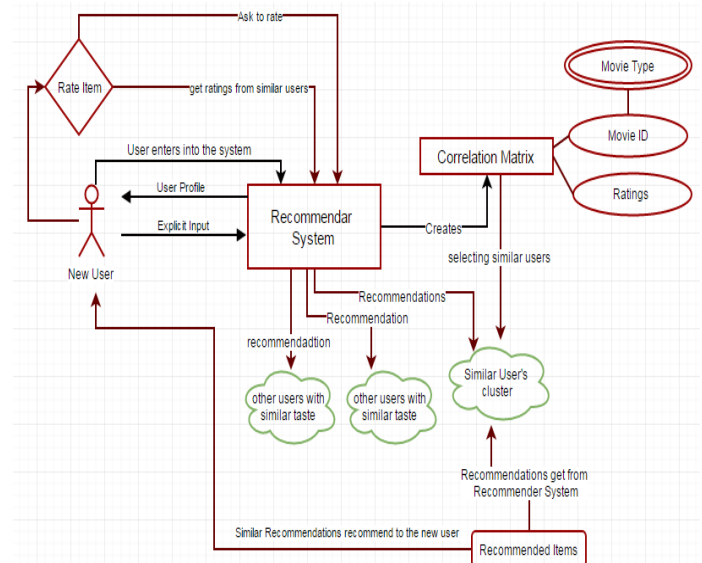


Figure 1: User Interaction with System Model

Our approach based on explicit information from the user to the system, first of all the user give his biography to the system and this stage the user is totally blind from the system techniques and strategies to get the proper feedback from the system and as the result from the explicit information the system returns his successful user's profile created into the system. The content-based approach can be used to get recommendations from the system.

3.2 Explicit Information

The Explicit feedback is an active action from the user to the system. This kind of feedback shows the activeness of the user and this is heavily depends on the willingness of the user to give feedback to the system. Explicit information are more

often preferred over implicit information, because of the accuracy is higher than the predicted or implicit information because this kind of information getting form directly to the user and user can enter the required information as his interest to the system to get accurate recommendations. The explicit information can be of different qualities. Users may give information not properly after coming back from a tiring day and not pay more attention. In a study by [12] they found that extreme explicit information is more consistent than realistic opinions. The consequence is that we cannot get a better and more reliable resolution of the user preference model just by increasing the probability of accuracy of start rating and predicted recommendations. The vast majority are using the extreme values. Another finding in their research was that similar items grouped together gave more consistent feedback and that fast ratings do not yield more inconsistencies. It is hard to ensure that users leave their explicit information. The laziness of the average user results in a relatively low ratio of information per user.

3.3 Calculation the Correlation Coefficient and Selecting Neighbors

Pearson correlation coefficient technique [13] is used to determine the correlation between the preferences of the user who is seeking a recommendation from other users as follows:

$$\rho_{xy} = \frac{\text{cov}(X,Y)}{\sigma_X\sigma_Y} = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2(Y_i - \bar{Y})^2}} \quad (5)$$

Where X is the user, who needs recommendations, \bar{X} is the average rating of X , X_i denotes the rating for the i^{th} item by user X , and other users represent by Y . "Figure 2" is the example that shows the Pearson correlation coefficient between U_x and U_1 is -1, and that between U_x and U_2 is 0.94. This indicates that U_2 has almost similar preferences to U_x . In the next step, neighbors are chosen using the results of (5). In this step, a correlation coefficient value close to 1 is first selected as the threshold value. Users with a correlation coefficient to U_x greater than this threshold value means 1 are selected as neighbors for the user.

Items\ Users	I1	I2	I3	I_m
U1	0	0	0	X
U2	0	0	0	0
U3	X	0	X	0
....
U_n	0	X	0	X

	U1	U2	U_n
U_x	-1	.94	-0.94

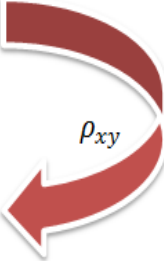


Figure 2: Calculation of Pearson Correlation Coefficient

3.4 Predicting Preferences

The final step is to predict preferences based on the ratings from neighbors as follow:

$$P = \bar{X} + \frac{\sum_{Y \in \text{raters}} (Y_n - \bar{Y}) \rho_{xy}}{\sum_{Y \in \text{raters}} |\rho_{xy}|} \quad (6)$$

Where \bar{X} the average rating for user is X , Y_n is the rating given by the other users for the n^{th} item, \bar{Y} is the average rating given by the neighbors of X for the current item. Finally, ρ_{xy} is the Pearson correlation coefficient between X and the other users Y . The raters are a set of users who input ratings for the item according to their interest. The result P in eq. (6) is the predicted value of an item for user X .

3.5 Clustering for Similar Users'

For clustering method we need three elements to make cluster. Nodes, edges and weight. Let suppose the user as the nodes, relationship between users as edges and the degree of similarity as weight. According to user's that's already exist in the system, extract their dynamic data, user-item matrix is described in "Table 2". With the ratings of items those are both watched by some two users, Pearson similarity is computed by (7):

$$r_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[n \sum x_i^2 - (\sum x_i)^2][n \sum y_i^2 - (\sum y_i)^2]}} \quad (7)$$

Where x and y means there are some two users, x_i and y_i shows the ratings of items from user x and user y respectively, and n is the number of items. The results are between -1 and 1, where only r_{xy} up to 0 indicates there is similarity between them. The Pearson weight larger means the stronger in relationship.

K-means [14] which is one of unsupervised learning algorithms in data mining is used to divide into sub-communities. A set of means $C = \{C_1, C_2, \dots, C_k\}$ is got by classifying a set of nodes $X = \{x_1, x_2, \dots, x_n\}$ which have n nodes, into k sub-communities. In first, the k means in C are initialized randomly. Secondly, each node x_i in X get its nearest mean c_j in C by computing from (8).

$$Y = \sum_{i=1}^n \min_{1 \leq j \leq k} (x_i - c_j) \quad (8)$$

Let suppose Y exists. Then all nodes in X are grouped into the sub-community $C_j (j = 1, 2, \dots, k)$ of the largest similarity, followed the central means of each sub-community as the new means. Above two steps are repeated until convergence is achieved or reach the maximum number of iterations.

3.6 Predicted Rating for Cold-User

To select the similar user to the new user, we can build user and item matrix using a user's information stored into the recommender system.

In Table 2, the items in each row are representing as movies and the column representing the user's. Thus, there are a total of m movies and n users in the matrix. There are two types of marks in the matrix one is "O" and another is "X" marks in the matrix denote that whether or not a specific user watched the movie, we can also say that, the O means that the user evaluated the movie and the X means that user did not. Thus, the total area marked in the matrix with an X is the sparse area. The first user we select a probe user to recommend items. In Table 2, user U_a is representing as probe user. We can extract the information from all the movies rated by the U_a , and randomly select a movie as a probe item that is used to predict

a movie recommendation. Then, we construct a sub-matrix to determine whether or not the user evaluated the movie.

Table 2: User and Item Matrix

Users \ Items	Items					
	I1	I2	I3	Im	
Ua	o	o	o	x	
U1	o	o	o	o	
U2	x	o	o	o	
.....	
Un	o	x	o	o	

In “Figure 3” the matrix on the left is same as the matrix we shown in the previous figure, and the matrix on the down side is the sub-matrix is used to determine the users other than the probe user evaluated the probe movie or not. In the “Figure 3” cold user is mentioned as user *Ua* and probe item is item *Im*. We can select the item that has no rating for probe user, so we can predict the rating of probe item. Thus, the user *Ua* has no rating for the item *Im*. We only include the users who only evaluated the probe movie that are in the rows of the sub-matrix. The columns in the sub-matrix are representing as items rated by the probe user, and an “O” means the users in the sub-matrix evaluated the same movie as the probe user while the “X” means they did not.

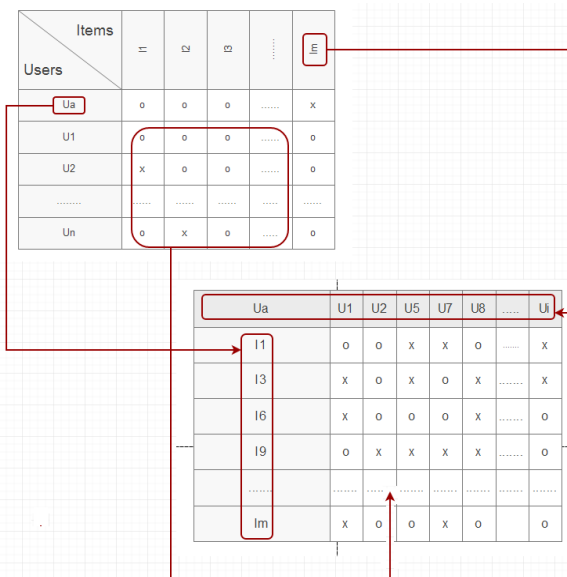


Figure 3: Cold User with sub matrix

In “Figure 4” the corresponding list for each column has their own. The elements in the list are the users who evaluated the movie. For example, the list for item I1 consists of users U1, U2 and U8. Likewise, the list for item I6 consists of users U2, U5, and U7. Specifically, users U1, U2 and U8 are those who have evaluated and representing an O for item I1, and users U2, U5, and U7 are those users who have an O for item I6, that’s mean that the users U1, U2 and U8 have a rating for item I1, and the users U2, U5, and U7 have a rating for item I2 Finally, we select similar users using the ratings given by the users on each list.

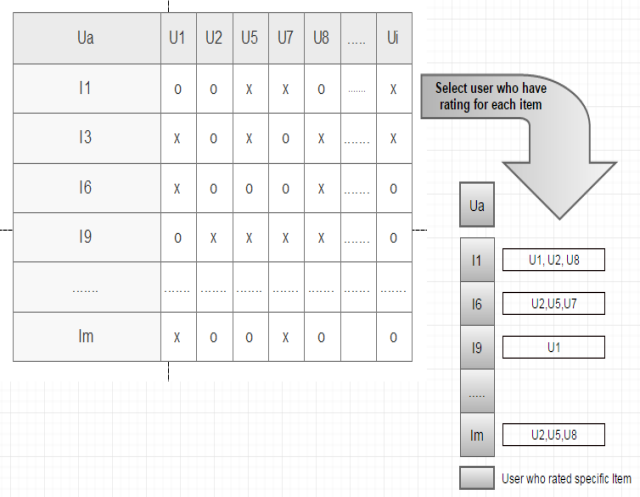


Figure 4: Extract User list for each item

In “Figure 5” is an example of the process for selecting similar users for item I1. For example, there are six total users who evaluated item I1: U1, U2, U8, U12, U17 and U27. The rating given by the probe user for item I1 was 4. The table on the right in “Figure 5” shows the ratings given by the six users. In this situation, users U1, U8, and U12 can be selected as similar users since rating of probe user is 4 and users U1, U8, and U12 are also 4 for the probe item.

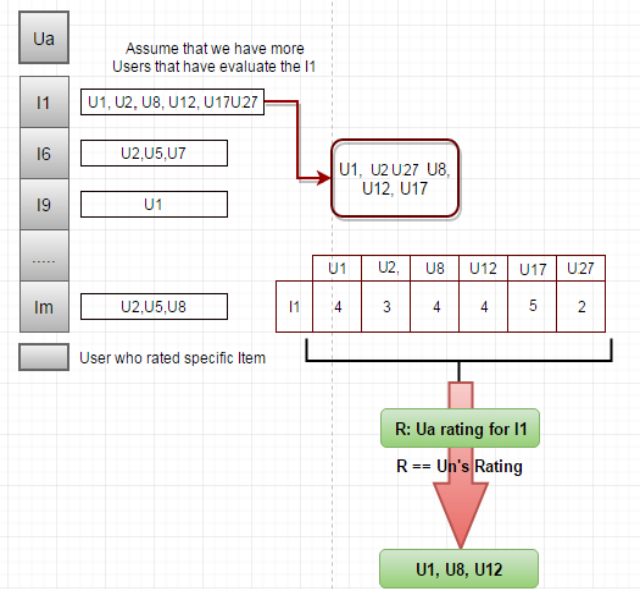


Figure 5: Predicting Rating for Cold User for Item I1

In summary, we first select the cold user. Then, we randomly select a cold item from the information stored in the system, and subsequently group items rated by the probe user. We extract users who evaluated an item at least once based on a list of items. Finally, we select similar users using ratings given by the extracted users and the probe user.

4. Experiment and Results

4.1 Movie-Lens Dataset and Data Modification

The Movie-Lens (<https://movielens.org/>) dataset will use in this paper for proposed experiment. There are multiple options for dataset available for research purposes with different strategies, but we are going to use dataset from Movie-Lens, there are 100,000 ratings and 6,100 tag applications applied to 10,329 movies by 668 users. And each person had rated at least

20 movies. Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. Before start our experiment the downloaded data needed to format in required condition. We used file movies.csv that contains movie id, title and genres fields. Genres field contain multiple of data in one column that we are going to use as user explicit preferences to reduce the cold-start problem. First of all we separate genres data in multiple columns with multiple genres filed. For example the data was in this form (Adventure|Animation|Children|Comedy|Fantasy) in a single genres field the given “Figure 6” is showing the actual form of data.

Row No.	movi...	title	genres
1	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	2	Jumanji (1995)	Adventure Children Fantasy
3	3	Grumpier Old Men (1995)	Comedy Romance
4	4	Waiting to Exhale (1995)	Comedy Drama Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action Crime Thriller
7	7	Sabrina (1995)	Comedy Romance
8	8	Tom and Huck (1995)	Adventure Children

Figure 6: Movies Data Representation

Then we change into multiple columns like genre1 contain “Adventure”, genre2 contain “Animation”, genre3 contain “Children” respectively. We had total 18 sub categories that we can divide our movies information data. Then we converted the polynomial data form into numeric data form to perform clustering approach. The “Figure 7” is showing the modified data form.

The representation is quite different because we use number 101 for category “Action”, 102 for category “Adventure” and so on with respect to 111 for “Horror” and 115 is for category “Sci-Fi”. The sign “?” is showing the missing data. For example movie named “Toy Story (1995)” belongs to the category “Adventure”, “Animation”, “Children” and so on. But this move does not belong to the category called “Horror” and “Sci-Fi”. So the Genres1 will contain only number 102, 103 104 respectively but does not contain 111 and 115 number codes.

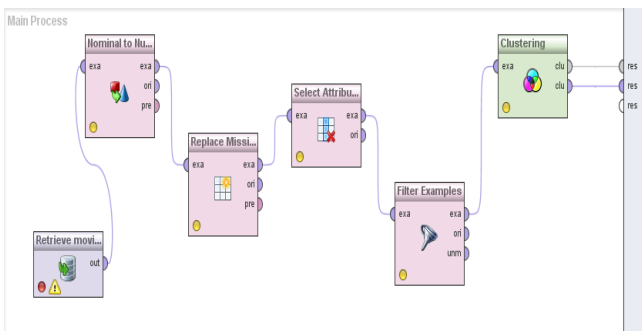


Figure 7: Clustering for Similar Users

Firstly we have input dataset from the modified movies.csv file then we have function that converts nominal values into numeric values for compatibility with K-mean algorithm. Replace missing function will handle the null values that are showing in “Figure 7”. We have select attribute function for selecting multiples of attribute as required from user explicit information from his given preferences. Then we have the

function filtering that perform filter actions on the selected data as user preferences and the output from filtering we can use as input for k-mean function for making clusters

With K value 3, we have 3 clusters that divided by movies id’s and every cluster have multiples of recommended users the more we have clusters the more we need efforts to select cluster that is more appropriate and required to the users preference. Cluster_1 contains the similar user with cold user explicit information.

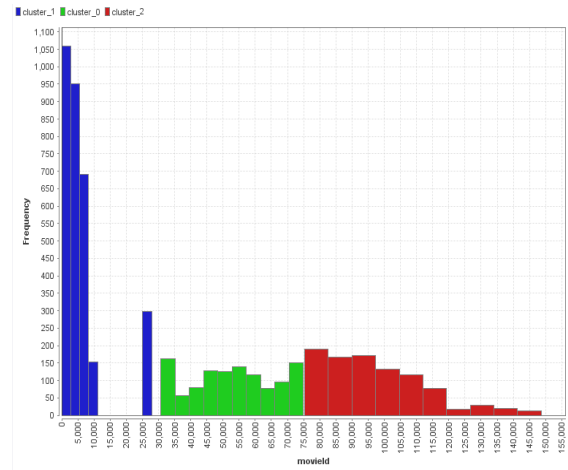


Figure 8: Clustering for Similar Users

4.2 User Explicit Preferences for Single Attribute

If users have only one preference that he likes only one kind of movie then system will recommend only one type of movies. For example if user like movies with category “Drama” then system recommend only those movies that are in category “Drama” and containing the number code 108 respectively. The following “Figure 9” is the recommendations from the system.

By considering cold user interested the category type “Drama” and he enters it as his explicit information to the recommender system. In the data-set “Drama” category that is denoted by number code 108. The one genres field contains multiple of movies categories with different number coding, the above table is the example of recommendations with items called movies representing with movies Id that are similar to the user information.

Row No.	id	cluster	movieid	genres1	genres2	genres3	genres4
2	2	cluster_1	11	105	108	114	114
3	3	cluster_1	14	108	110	112	114
4	4	cluster_1	16	106	108	112	114
5	5	cluster_1	17	108	114	112	114
6	6	cluster_1	20	101	105	106	108
7	7	cluster_1	22	106	108	111	113
8	8	cluster_1	24	108	115	112	114
9	9	cluster_1	25	108	114	112	114
10	10	cluster_1	26	108	110	112	114
11	11	cluster_1	27	104	108	112	114
12	12	cluster_1	28	108	114	112	114
13	13	cluster_1	29	102	108	109	113
14	14	cluster_1	30	106	108	112	114
15	15	cluster_1	31	108	110	112	114
16	16	cluster_1	34	104	108	112	114

Every row contain value 108 only one time and not repeating but every column can contain multiples of value 108

Figure 9: Recommended Items with Single Attribute

In the real world example a typical user normally like multiples of categories of movies not only a single category type of movies and with the passage of time his interest might be change with other categories. If we use single attribute

technique for recommendations then we have recommendations in thousands and repeated.

4.3 User Explicit Preferences as Multiple Attributes

We can assume that a cold user when he enters to the recommender system he can have a multiple options to select for his explicit information to the system to get more accurate recommendations as the feedback from the recommender system.

- Action → 101
- Crime → 106
- Horror → 111
- Mystery → 113
- Thriller → 116

4.4 Mathematical Set Theory Approach

We can have multiple explicit information tan we can assume as objects for a set. We can also elaborate as {101}, {106}, {113}, {111}, and {116}. These are all single attributes set that we already discussed in previous section. Now we have another and enrich approach based on Mathematical set theory. We can take these sets as the subset and if we apply the reverse strategy from subsets to get the actual set can be possible and that set can have the multiple objects and from that set we can also obtain the multiple subsets.

A= {101}, B= {106}, C= {113}, D= {111}, and E= {116}. If we can combine these subsets into a single superset then we have the situation like this: U = {{101}, {106}, {113}, {111}, {116}} or U= {101, 106, 113, 111, 116} respectively.

$$U \in A, B, C, D, E$$

$$A \subseteq U, B \subseteq U, C \subseteq U, D \subseteq U, \text{ and } E \subseteq U$$

We can again convert into multiple subsets.

U = {101, 106, 111, 113, 116}, we can call it super set.
 U={{},{101},{106},{113},{111},{116},{101,106},{101,113},{101,111},{101,116},{106,113},{106,111},{106,116},{113,111},{113,116},{111,116},{101,106,11},{101,106,111},{111,106,116},{101,111,113},{101,113,116},{111,113,116},{101,106,111,113},{101,106,113,116},{106,111,113,116}}

The total number of subsets for the superset can be explaining by the mathematical term:

$$U = n \times n, \quad n \in \text{objects} \quad (9)$$

Row No.	id	cluster	movieid	genres1	genres2	genres3	genres4	genres5
1	1	cluster_0	22	106	108	111	113	116
2	2	cluster_0	366	108	111	113	116	0
3	3	cluster_0	1327	108	111	113	116	0
4	4	cluster_0	1350	111	113	116	0	0
5	5	cluster_0	1407	105	111	113	116	0
6	6	cluster_0	1644	111	113	116	0	0
7	7	cluster_0	1717	105	111	113	116	0
8	8	cluster_0	1974	111	113	116	0	0
9	9	cluster_0	2232	111	113	115	116	0
10	10	cluster_0	2338	111	113	116	0	0
11	11	cluster_0	2810	103	111	113	116	0
12	12	cluster_0	2841	111	113	116	0	0
13	13	cluster_0	2902	111	113	116	0	0
14	14	cluster_0	3113	101	109	111	113	116

Figure 10: Recommended Items with Single Attribute

After superset the largest subset is the first preference as input for the system to get recommendation. If there is no possible

recommendation then largest other possible attributes combination consider as input for the system. Consider we have input with three objects {111,113,116} from user explicit information, this input belongs to the categories Horror, Mystery and Thriller.

There are total 112 recommended movies out of 10,000 movies that can show that the more we have big set the more we have accurate and less recommendations.

4.5 Performance and Evaluation

Since the database movie-lens does not take into account cold-start users (users with less than 20 votes), the strategy we use we have removed votes of this database in order to achieve cold-start users and make this situation. Indeed, we have removed randomly between 5 and 20 votes of those users who have rated between 20 and 30 items. In this way, those users who now result to rate between 2 and 20 items are regarded as cold-start users. We recover the removing votes of those users with greater than 20 votes despite of removing some their votes (in this way, these users keep immutable in the database). In order to estimate the performance of the proposed approach, the precision and recall strategy can be used.

Recall: The recall score is the inversely proportion of items from test set that appear among TopN of the ranked list from the training set. This measure should be as high as possible for good performance. Assume N is the number of items which are in the testing set and liked by users, n is the amount of items which the new user likes and appears in the recommended list. So, the recall is computed as follows:

$$\text{Recall} = \frac{n}{N} \quad (10)$$

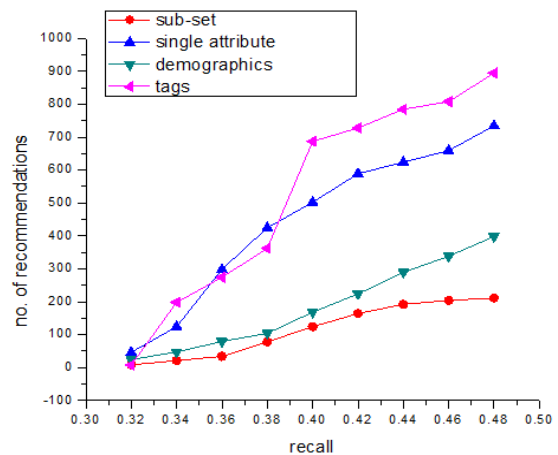


Figure 11: Recall Validation Cold-start User

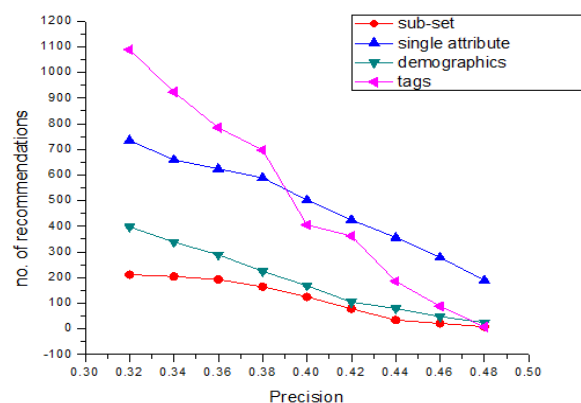


Figure 12: Precision Validation Cold-start User

Precision: The precision is the proportion of recommended items that the new users actually liked in the test set. This measure is also as high as possible for good performance. The precision is computed as follows:

$$\text{Precision} = \frac{n}{\text{Top}N} \quad (11)$$

Evolution of the results of results of tags, single attribute, demographic and sub-set (proposed approach). Precision and recall throughout the range of number of recommendations $N = \{0, \dots, 1000\}$. In this experiment we make use of all the cold-start users (no more than 20 votes.)

The content-based filtering algorithm, after applying the linear regression the every recommended movies contain a predicted rating, which also enables us to evaluate the difference between the predicted and the actual user rating, once a user rates a movie. This evaluation is measured by the mean absolute error (MAE). This error is defined as the absolute value of predicted rating, p , subtracted by the actual user rating, a . The mean of these errors is the MAE and explains how far the algorithm is from the optimal predictions.

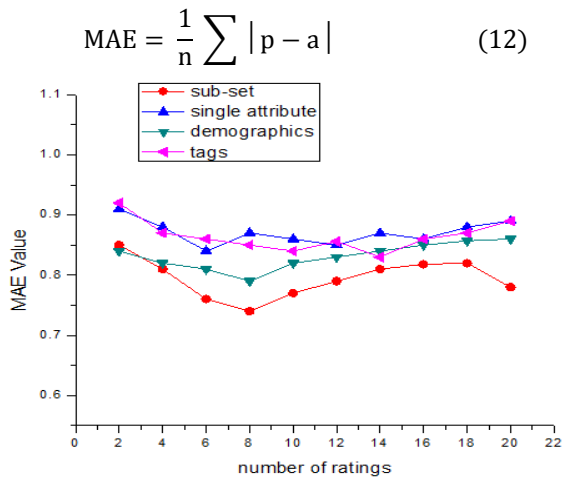


Figure 13: MAE values determined using Mathematical Approach and traditional method

References

- [1] P. Shardanand and U. Maes, "Soacial information Filtering Algorithm for Automating word of mouth," In Conference on Human Factor in Computing System, vol. 95. 1995.
- [2] J. A. Konstan et al., "GroupLens: applying Collaborative Filtering to Unset News". In Communication of the ACM, Vol. 40, pp. 77-78, 2004.
- [3] N. Stah, Y. Wang, P. Gorgels, A. L. Rutledge and L. Aroyo, "CHIP demonstration: Sementic-driven recommendations and museum tour generation," in Sementic Web Challenge, Busan, Korea, 2007.
- [4] N. Mohammadhossein, K. B. Fard A. H. Nabizadeh Rafsanjani, N. Salim, "New Recommendation System Model Based on Semantic Similarity in Movie Domain," Journal of Basic and Applied Scintific Research, vol. 3(7), pp. 258-273, 2013.
- [5] N. Izumi, T. Morita, and T. Yamaguchi M. Ishikawa P. Geczy, "Information diffusion Approach to Cold-start Problem," In Web Intelligence/IAT Workshop, pp. 129-132, 2012.
- [6] M. J Pazzani and D Billsus, "Content-based Recommendation Systems," The Adaptive Web Springer, pp. 325-341, 2011.
- [7] F. Ortega, A. Hernando, J. Bernal J. Bobadilla, "A Collaborative filtering approach to mitigate the new user Cold-start problem," in Knowledge-Based Systems , vol. 26, pp. 225–238, 2012.
- [8] S.C. Chan, F.L. Chung C.W. Leung, "An empirical study of a cross-level association rule mining approach to cold-start recommendations," in Knowledge Based Systems, vol. 21(7), pp. 515–529, 2008.
- [9] I. Ha, and G.S. Jo H.N. Kim A.T. Ji, "Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation," in Electronic Commerce Research and Applications, vol. 9(1), pp, 73–83, 2010.
- [10] Le Hoang Son, "A hybrid user-based fuzzy collaborative filtering method in Recommender Systems," in Expert Systems with Applications, vol. 41(15), pp. 6861–6870, 2014.
- [11] P. Massa and B. Bhattacharjee, "Using trust in recommender systems: anexperimental analysis," Trust Management, Springer, pp. 221–235, 2009.
- [12] N. Oliver X. Amatriain J. M. Pujol, "I like it. i like it not: Evaluating user ratings noise in recommender systems," User modeling, adaptation, and personalization, Springer, pp. 247–258, 2009.
- [13] S. M. Choi, S. K. Ko, Y. S. Han B. Scholz, "A Recommendation System Based on a Subset of Raters," In 6th International Conference on Ubiquitous Information Management and Communication, 2012.
- [14] B Lopez, and J. L. D. L Rosa M. Montaner, "A texonomy of Recommender Agent on the Internet," in Artificial Intelligence Review, vol. 19, pp. 285-330, 2013.

Author Profile

Haider Khalid received the Bachelor of Science and Master of Science in Information Technology degrees from University of the Punjab, Lahore Pakistan in 2010 and 2012, respectively. During 2012-2013, he did job as Software Engineer. In September 2013, he moved to China for his 2nd Master's degree in Software Engineering in Jiangsu University on Chinese Scholarship Council (CSC). During his studies in China, his research field is Recommendation Systems. He will continue his studies in Jiangsu University and start his PhD from September 2016.



Shengli Wu received his Master's and PhD degrees from Southeast University, Nanjing China in 1989 and 1996, respectively. He is Professor in Jiangsu University and has been doing teaching and research in the area of data base and information systems for long time. He has been involved in many research projects in China and abroad. Before he came back to China

in June 2012, he had been worked in UK universities for over a decade. His major research areas are: Data base and information systems information retrieval, machine learning and web technology.