User Classification in a Personalized Recommender System using Learning Automata

Mansoureh Ghiasabadi Farahani[1] , Akbar Torkestani* [2] , Mohsen Rahmani [3]

[1] Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, mansoureh_ghias_stu@qom-iau.ac.ir
[2] Department of Computer Engineering, Arak Branch, Islamic Azad University, Arak, Iran, j-akbari@iau-arak.ac.ir
[3] Department of Computer Engineering, Faculty of Engineering, Arak University, rahmanimohsenir@hotmail.com

Abstract

The recommender systems are the popular personalization tool for helping users to find pertinent information based on preferences kept in individual profiles. A user profile plays an essential role in the success of the recommendation processes, so the recommender systems must design a profile to identify the user's needs. The accuracy of the user profile affects the overall performance of the recommender system. Personalizing through creating a user profile is considered a challenge because people's interests are changing over time. In this paper, a learning method is proposed that uses user feedback to improve the accuracy and precision of the recommendation list. This method utilizes learning automata to complete the user profile. The user preferences represent in the form of a weight vector. This vector is the action probability vector of the learning automata; it is updated according to user feedback. Experimental results based on Movie Lens 100k, Movie Lens 1M, and Netflix datasets show that the proposed approach is superior to existing alternatives.

Keywords: Recommender system, Content-based filtering, Change of user interest, Adaptive user's profile
Article history: Received 03-Nov-2021; Revised 02-Dec-2021; Accepted 07-Dec-2021. Article Type: Research Paper

https://dorl.net/dor/20.1001.1.22519246.2021.10.04.10.5

1. Introduction

Recommendation systems are used to predict user preferences and help the user search and select the services or products. Providing personalized recommendations is often problematic because of the information overload problem. The methods of providing personalized recommendations are generally based on a user profile. The user profile estimates the user preferences and interests of a particular domain. User profile plays a crucial role in recommendation systems; in fact, the user profile accuracy in showing the user preferences will seriously affect the efficiency of the recommendation process.

An adaptive and personalized recommender system makes and keeps a model of user and data between humans and computers. The user profile is the main component of every interactive system since it necessitates having all the necessary information, which is significant to adapt and personalize the user. The purpose of personalized systems is to meet the user's needs without their explicit expression. So, there is a problem of learning user preferences in many applications like personalized information retrieval and personalized recommender systems. The user profile is a personal file of preferences and tastes. Since the initial profile of the user is incomplete and vague, using methods to update the information of the user profile is necessary. The more accurate the user profile, the services will undoubtedly be more satisfactory for the user [1-3]. A user profile can be static or dynamic, whereas the user preference is dynamic and temporal. Therefore, making a dynamic profile for the users would be more reasonable. The research gap in personalized recommender systems is non-tracking the change of users' interests during time automatically.

In this study, a content-based recommendation algorithm is suggested, called LARC, to resolve the problem of user interest change using the information of both movies' features and user ratings. The feature vector of items forms the user interest vector. The user interest weight vector is updated based on the user rating in each user

interaction with a recommender system. This vector is defined as an action probably vector of learning automata, which helps make a dynamic user profile to personalize recommendations. As LARC goes on, the choice probability of a feature for being a part of the user's interests converges to its degree of importance. The learning automaton is a suitable model for solving the above problem because of has the following features:

(1) The learning automata can perfectly adapt itself to environmental changes. This feature is very suitable for recommender systems because user preferences change over time.

(2) The learning automata complete the information required for decision-making in an iterative process based on the reinforcement signal received from the environment. The proposed algorithm uses this feature to complete the user's profile.

The suggested method is tested on three datasets Movie Lens 100k, Movie Lens 1M, and Netflix Prize. The obtained results are compared with IBGCRS, classical content-based, DTI-CF, GC-MC, and SCoR. The results reveal the supremacy of the suggested approach regarding precision, recall, mean average precision, mean average recall, and accuracy.

The rest of the article is as follows: the following section reviews related works. Section 3 briefly studies the learning automata theory. In section 4, the recommender system algorithm is provided based on learning automata. The evaluation metrics, dataset, compared strategy are explained in Section 5. Section 6 displays the performance of the proposed approach through simulation experiments and its comparison with the previous methods, and finally, the conclusion is provided in section 7.

2. Related works

The two main challenges in creating user profiles are generating the initial user profile for a new user and constantly updating user profile information to match changes in user preferences, needs, and interests [4]. The majority of the recommendation algorithms to solve these challenges are approximately categorized into three types: collaborative filtering, content-based filtering, and hybrid filtering.

The collaborative approach recommends items that have been probably preferred by target users based on collective intelligence or rating by other users with attitudes similar to the target user. Two conventional approaches in this method are Item-Based Collaborative Filtering and User-Based Collaborative Filtering. Item-Based Collaborative Filtering predicts similar items in each group and generates a list of similar items based on rating, while User-Based Collaborative Filtering calculates the similarity among users based on user rating data. This approach works excellent in the face of items that are difficult to analyze using only explicit information. Collaborative Filtering approaches often have two problems: data sparsity and cold start. Although a massive number of items are available, each user usually rates a few items. Therefore, it is not easy for a recommender system to measure user similarities precisely based on a limited number of ratings. In addition, when a new user enters the system, there is usually inadequate data for interpreting the user's preference precisely [5-7]. Papadakis et al. [8] borrowed the idea of the Vivaldi algorithm for internet latency estimation, which was modified and expanded to be used for recommendations. In the modified version of Vivaldi, the user nodes and item nodes are considered replacements for network nodes. Also, distances estimated by user ratings on items are used instead of latencies. The system convergence, which means obtaining the desired position for each node, the Euclidean distance between any pair of nodes, provides an accurate prediction of recommendations. The heterogeneous evolutionary clustering collaborative filtering prediction rating method is proposed in [9]. At first, a network is built that its nodes are users and items. The evolutionary clustering algorithm is created for heterogeneous network nodes. Users based on the created network model update the states. Depending on their stable states, users who have similar state values are located in the same clusters. Eventually, according to the user-based collaborative filtering method, the mean score for the active user in the prediction formula is adopted in each cluster. Chen et al. [10] proposed a dynamic clustering recommender algorithm according to the time-weighted similarity matrix. The period is divided into two parts in this approach, and different periods process different network structures. It constructs the user's interest signed network model for each period. An improved similarity measure is used to calculate users' interest similarity. Based on the impact of rating times on recommendation results, this method applies two similarity matrixes created from consecutive periods. Then, the users and items are clustered based on an evolutionary network model. Finally, the proposed approach recommends items by applying the collaborative filtering method in each cluster. Embedding, which transforms users and items into vectorized representations, is the main component in learnable collaborative filtering models. Different co-occurrence information, including co-disliked item-item co-occurrences and user-user co-occurrences, extracted from the user-item interaction matrix, are used by Tran et al. [11].

They proposed a joint model combining Weighted Matrix Factorization, co-liked embedding, co-disliked embedding, and user embedding. A graph-based auto-encoder framework is proposed by Berg et al. [12] for matrix completion. The latent features of user and item nodes are generated by the auto-encoder using a form of message passing on the bipartite interaction graph. These latent user and item representations are applied to reconstruct the rating links through a bilinear decoder. A new framework of neural graph collaborative filtering is proposed by Wang et al. [13] to complete the user-item interactions and, more specifically, the bipartite graph structure into the embedding process. This approach leverages high-order links in the graph of user-item integration and explicitly incorporates collaborative signals into the embedding function of the model-based collaborative method.

In content-based filtering, user profiles and item profiles are created based on items that have already been selected and based on keywords and item features. This method recommends items whose profiles are similar to the target user's profile. Therefore, the recommended items are similar to the users' previous choices. The problem with this method is that the user profile is commonly incomplete because users do not like to express their interests explicitly [14-16].

Hybrid filtering is a combination of collaborative filtering and content-based filtering. This method is a complementary approach so that each method solves its problems with the strengths of the other method [17-20]. Symeonidis et al. [20] have suggested a user profile based on weighted features to reveal a similarity between users and items features. This content-based user profile is a hybrid approach made using collaborative and content-based filtering. First, a user model is made according to the feature weight showing the similarity between users and features. Then, based on the recovered information from the weighing plan of Term Frequency Inverse Document Frequency (TFIDF) in collaborative filtering, they have provided an algorithm based on feature frequency to generate a list of top N recommendations. In [21] has been designed a framework to find a better way to show the user preferences and aggregation of domain features in a recommendation system. This method applies item domain features to build a user preferences model to provide recommendations using collaborative filtering on these personalized domain models. Reference [22] suggested a hybrid approach. This approach defines a new user model that is known as a user-feature model. In this model, the user preferences are modeled based on item features and user ratings. The user-feature model has been made using Fuzzy C Mean on the user-item model. In [23], a multi-purpose and multi-level user profile has been suggested to improve the efficiency of personal search. The user profile includes different kinds of resource features, each of which shows if the user is interested in the feature or not. All approaches mentioned above design users' profiles as static; therefore, they have not considered the changes of user interests; as a result, recommendations of the traditional recommendation algorithms often do not satisfy users.

Reference [24] has created a dynamic user profile that combines a demographic profile and an individual profile based on the number of events created by the user. This model gradually fades the general components and reinforces the individual components. These components are genre frequency and user ranking, respectively. In [25], a user profile for personalization recommendations has been designed in which the items have multiple features (for example, genre features, keywords, director and…), and user preferences for the items have been demonstrated in the form of explicit feedback, or they have been obtained through user implicit feedback. The exclusive weights of vectors that compose the user model will be calculated dynamically based on the user's behavior and the other users' behavior. This user model is used in every domain of which metadata can describe the items. Jing Li et al. [26] have provided a hybrid recommendation algorithm for solving the problem of data sparsity and change of user interest. The user interest vector combines the user rating matrix and items feature vector, which is updated gradually. Then, the user similarity matrix is built based on the interest vector and the user-rating matrix. In this approach, data sparsity has a minimum effect on the results. Tae-Gyu Hwang et al. [27] have provided an algorithm for describing a movie, which uses movie genre and movie rating to increase the accuracy of rating predictions. The suggested algorithm measures the numerical correlation between movie genres based on rating information; it then categorizes the movies using the numerical correlation of genres and generates a recommendation list for the user. Finally, the movie's rating on the recommended list will be predicted using the collaborative filtering method. The reference [28] tried to solve the problem of data sparsity and the problem of cold start. In this paper, a hybrid collaborative filtering method has been suggested, including two parts: collaborative filtering based on the genre that manages the problem of cold start of new items and provides some recommendations for users despite the low number of ratings. The second part uses the user-based collaborative filtering algorithm. Sebastian Fremal and Fabian Lecron [29] have presented a clustering approach based on the item's metadata

information. The items are clustered based on their genres. As the items may have some genres, they are placed in several clusters. Each cluster represents its predicted rating, and then, these ratings are combined using weighting strategies. The significant innovation of reference [30] is that the suggested approach changes the item rating to the feature ratings, and then, modeling is completely done at the feature level. Hence, this approach uses the benefits of collaborative and content-based filtering. For example, even if the users do not rate the same items, the features of items can be shared, which helps to solve the problem of the cold start of a new item. Since the final prediction of item rating was achieved by merging its features' ratings, the suggested model is not involved items, and this is the fundamental difference between the suggested approach and the other modeling methods of feature-based predictions. In reference [31], content-based filtering has been combined with collaborative filtering to achieve accurate recommendations. In this method, a graph-based model has been used to show the content of item features and rating information, which guides the user in selecting essential attributes of items and relates the users to the exciting items. These approaches use a dynamic profile, but those are not accurate in predicting the ratings for creating the recommendation list. Existing methods do not automatically track changes in users' interests and do not periodically update them.

The papers reviewed below use learning automata to provide web personalization. Web personalization is a set of operations that organizes the web experience for an individual user or a set of users. Since the users do not tend to express their features and interests, little information is generally available about users or items in the recommendation system. Forsati et al. [32] have used distributed learning automata to learn the previous users' behavior in observing the pages and clustering the pages based on the learned pattern that led to recommendation quality improvement. Talebbeigi et al. have provided a hybrid web recommendation system based on asymmetric cellular learning automata with multiple learning automata in each cell. This method tries to identify the information needed by the user and recommends the appropriate pages for users. In addition, in another article, a dynamic web system is proposed based on asymmetric cellular learning automata. The learning automaton interacts with users continuously and learns their behavior [22, 33]. Also, to overcome two challenges of cold start and data sparsity, Ghavipour et al. have proposed a continuous action-set learning automata-based method to adjust membership functions of fuzzy trust and distrust during the lifetime of the recommender system in terms of recommendation error [34].

3. Learning Automata

A learning automaton [35, 36] can boost its performance through learning to select its optimal action from a finite set of allowed actions obtained by iterative interactions with a random environment. The probability distribution of the action-set is the basis for the learning automaton to select an action randomly. The learning automaton uses the selected action as the input to the random environment. The environment uses a reinforcement signal to respond to the adopted action. The vector of action probability is updated using reinforcement feedback. The learning automaton tries to find the optimal action from the action-set for minimizing the average penalty from the environment. In systems where complete information is not available about the environment, learning automata can be useful [37]. Also, it can perform very well in dynamic, complex, and random environments full of uncertainties. In case of facing many hard-to-solve problems, utilizing a group of collaborative learning automata can also be useful.

A triple $E\equiv\{\alpha, \beta, \gamma\}$ can describe an environment, where $\alpha\equiv\{\alpha_1, \alpha_2,\ldots,\alpha_r\}$ represents the finite set of the inputs, $\beta\equiv\{\beta 1, \beta 2,\ldots, \beta m\}$ denotes the set of the values that can be taken by the reinforcement signal, and $\gamma\equiv\{\gamma_1, \gamma_2, \ldots, \gamma_r\}$ signifies the set of the penalty probabilities, where the element γ_i is linked with action α_i. If the penalty probabilities are constant, the random environment is called a stationary random environment, and if they vary over time, the environment is named a non-stationary environment. The environments according to the nature of the reinforcement signal β can be classified into P-model, Q-model, and S-model. In P-model environments, the reinforcement signal can only take two binary values of 0 and 1. Another class of the environment allows a finite number of the values in the interval [0, 1] can be taken by the reinforcement signal. Such an environment is called the Q-model environment. In S-model environments, the reinforcement signal lies in the interval *a*-*b*.

Learning automata can be classified into two main categories fixed structure learning automata and variable structure learning automata [38]. Variable structure learning automata are shown by a triple $\{\beta, \alpha, T\}$ where β is the set of inputs, α is the set of actions, and T is a learning algorithm. The learning algorithm is a recurrence relation applied for modifying the action probability vector. Let $\alpha_i(n)\in\alpha$, and $p(n)$ signify the action chosen by the learning automaton and the probability vector defined over the action set at instant *n*, respectively.

Let a and b signify the reward and penalty parameters. They determine the penalization of increases and decreases of the action probabilities in the respective order. Let r be the size of the actions vector that can be taken by learning automaton. At each instant n, the action probability vector $p(n)$ is updated by the linear learning algorithm given in (1), if the environment rewards the selected action $\alpha_i(n)$, and it is updated as given in (2) if the taken action is penalized.

$$\begin{cases} P_i(n+1) = P_i(n) + a[1 - P_i(n)] & j = i \\ P_j(n+1) = (1-a)P_j(n) & \forall j \neq i \end{cases} \quad (1)$$

$$\begin{cases} P_i(n+1) = (1-b)P_i(n) & j = i \\ P_j(n+1) = b/(r-1) + (1-b)P_j(n) & \forall j \neq i \end{cases} \quad (2)$$

If $a = b$, the recurrence (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a>>b$, the given equations are called linear reward-penalty (L_{R-P}), and finally, if $b = 0$, they are named linear reward-Inaction (L_{R-I}). In the last case, the action probability vectors remain unchanged when the environment penalizes the taken action.

4. The proposed approach

For the recommender system to be able to recommend suitable items to the user, it should know the interest of users. Since the user interest changes over time, therefore, tracing these changes is essential. This study aims to present an adaptive method using learning automata for finding the best-personalized recommendation list of the items so that the items listed in the final list will be the most relevant items according to the user preferences. The learning automata applied explicit user feedbacks to learn user interests and preferences. For providing satisfactory background to understand the suggested learning algorithm, some preliminaries are presented first. Consequently, this part is structured as follows: we introduce the features vector of an item and the user interest vector in sections 4.1 and 4.2, respectively. User rating prediction is discussed in section 4.3, and the learning method is introduced in section 4.4.

4.1. Features vector of item

We define an item by a set of features that describes its content. Since user feedback on each item demonstrates his/her interest in the features of that item. So we use features of items to make a user model. Each movie belongs to one or more genres (action, comedy, drama, …) available in datasets. The user rating on a movie shows his/her interest in the genres of the movie. This information enables us to achieve more details about user preferences.

In this paper, the movie set is denoted as $M=\{m_1, m_2, m_3, \ldots, m_n\}$ which n is the number of movies, the feature vector of each movie is indicated as $f_{mj}=\{w_{mj,1}, w_{mj,2}, \ldots, w_{mj,d}\}$, where $w_{mj,i}$ denotes the weight of the feature g_i in the movie mj. g_i is the feature of the movie. d is the length of the vector. j and i denote the movies and the feature, respectively. The feature vector f_{mj} of the movie m_j can be initialized via (3), Where the weight $w_{mj,i}$ will be 1 if the corresponding movie has the feature gi. Otherwise, it will be 0.

$$w_{m_{ji}} = \begin{cases} 1 & g_i \in m_j \\ 0 & g_i \notin m_j \end{cases} \quad (3)$$

4.2. The user interest vector

In this paper, we want to construct the user interest vector based on the feature vector of movies. The learning automata updates its action probability vector to find the weight of these features based on user rating to the movie.

In this paper, the user set is shown by $U= \{u_1, u_2, u_3, \ldots, u_m\}$, which m is the number of users. Each user has an interest vector $ui=\{g_1,g_2,\ldots,g_L\}$ with weight vector $f_{uj}=\{w_{uj,1}, w_{uj,2}, \ldots, w_{uj,l}\}$ in which L is the length of user preferences vector and $w_{uj,1}$ is the weight of i^{th} interest of user u_j. As mentioned in section 3.1, d is the vector length of movie features, which $d=L$.

4.3. Rating prediction

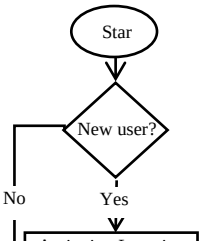
The proposed approach predicts the estimated rating for a candidate movie m_j by a user u_i as follows:

$$\hat{r}_{u_{i,j}} = \gamma * \bar{r}_j + \lambda * \bar{r}_{u_i} \quad (4)$$

Where $\bar{r}_j$ is the average rating given by other users on movie m_j, $\bar{r}_{u_i}$denotes the average rating given by ui on movies whose features are similar to features of movie m_j. Coefficients λ and γ are adjusted to obtain a better result.

4.4. Learning method

The suggested user profile learning method is defined as an adaptive problem that can be shown by a quintuple $<u_i, A_i, m_j, r_{i,j}, f_k>$. The learning automata A_i is assigned the user u_i. The $m_j \in M$ is the item recommended to the user u_i. The $r_{i,j}$ is the given rate by the user u_i to the movie m_j with the feature f_k. This learning automaton adjusted the weight of each user's interest. The flowchart of the proposed algorithm is shown in Fig. 1.



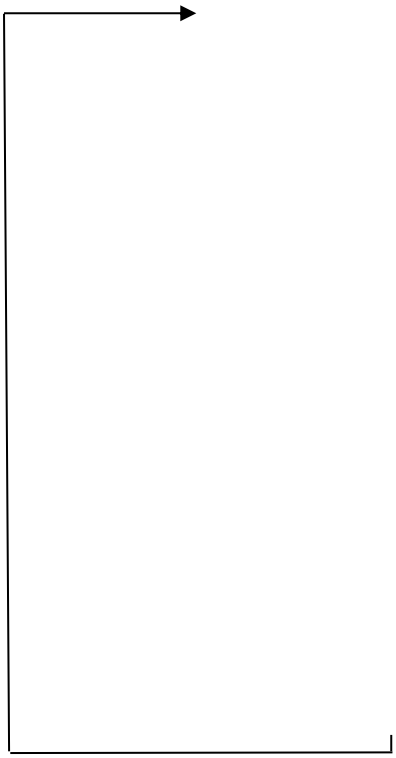


Fig. 1. The flowchart of the proposed.

In the suggested approach, a learning automaton A_i is assigned to the user u_i. The action set of A_i is considered the interest vector of the user, and the action probability vector of A_i is the preferences weight vector of the user u_i. The structure of the action set and the action probability vector are configured, as discussed in Section 4.2. At first, all actions have the same choice probability equal to *1/d* (*d* is the length of the user interest vector); this means that the user preferences are initially considered the equal proportions; this is because the learning automaton has no a priori information about the user interests. LARC adjusts the weight of the user's interests based on user feedback. The user feedback is an explicit rating that the user gives to the recommended item. Generally, the item's features are relevant to the user's interests if the user gives a high rating to the item. It is irrelevant otherwise. Fig. 2 shows the pseudo-code of the proposed approach.

The number of stages equals the number of items that are listed in the recommendation list. In each stage, one movie is recommended to the user in the following steps:

Step 1: Learning automaton A_i randomly chooses an action $\alpha_i^j \in \alpha$, according to the action probability vector.

Step 2: Movies that have not been recommended yet to the user u_i, and their j^{th} weight feature based on selected action is 1, are chosen as candidate movies.

Step 3: The rating prediction of these movies is calculated with (4).

Step 4: A movie with the highest rating prediction m_k is recommended to the user u_i.

Step 5: The user u_i rates movie m_k with rating r_{ik}.

Step 6: If $r_{ik}>=3$ then $\forall j\ f_{mk,j}=1$ then $\alpha_i^j \in \alpha_i$ is rewarded, and the other actions are penalized with (1). Otherwise $\forall j\ f_{mk,j}=0$ then $\alpha_i^j \in \alpha_i$ is penalized, and the other actions are rewarded with (2). This step aims at training the learning automaton by updating its action probability vector to find the user's interest with the highest probability.

These steps are repeated to complete the recommendation list.

5. Experiment

In this section, we first discuss the metrics used for evaluation and the dataset used for the experimental work. Next, the compared baselines with the proposed LARC model are presented briefly.

5.1. Evaluation metrics

The users' confidence in a specific recommender system does not depend directly on the set of possible predictions accuracy; however, this is achieved when the user agrees with a reduced set of recommendations made by the recommender system. Therefore, in this paper, accuracy measurement metrics in addition to quality evaluation criteria are also considered. The considered metrics in this paper are 1) precision, which shows the ratio of related recommended movies from the total number of recommended movies, 2) recall, which indicates the ratio of relevant recommended movies from the number of relevant movies, 3) Root Mean of Square Error (RMSE), 4) Mean Absolute Error (MAE) [5].

$$RMSE = \left(\frac{1}{\#u}\right) * \sum_{u \in U} \sqrt{\left(\frac{1}{\#Z_u}\right) * \sum_{i \in Z_u} \left(\hat{r}_{u,i} - r_{u,i}\right)^2} \quad (5)$$

$$MAE = \left(\frac{1}{\#u}\right) * \sum_{u \in U} \left(\left(\frac{1}{\#Z_u}\right) * \sum_{i \in Z_u} \left|\hat{r}_{u,i} - r_{u,i}\right|\right) \quad (6)$$

$$Precision = \left(\frac{1}{\#u}\right) * \sum_{u \in U} \#\{i \in Z_u | r_{u,i} \geq \theta\}/n \quad (7)$$

$$Recall = \left(\frac{1}{\#u}\right) * \sum_{u \in U} \frac{i \in Z\_u \,|r\_(u\#\{,i) \geq \theta\}}{\#\{i \in Z_u | r_{u,i} \geq \theta\} + \#\{i \in Z_u^c | r_{u,i} \geq \theta\}} \quad (8)$$

Where, U is the set of the users, $r_{u,i}$ denotes the rating of user u on item i, $\hat{r}_{u,i}$ denotes the predicted rating of user u on item i, X_u is the set of recommendations to user u, and Z_u is the set of n recommendations to the user u. θ is the accepted rating threshold provided by the user u, to the item i, and $Z_u^c = X_u - Z_u$, and # shows the number of set members.

5.2. Datasets

We conducted our experiments on three well-known datasets, two Movie Lens datasets used by the Group Lens research project of Minnesota University, and the Netflix Prize dataset. They are standard datasets often applied in recommendation system research: 1) the Movie Lens 100K dataset consisting of 100,000 ratings from 943 users on 1,682 movies, denoted as ML-100K and 2) the Movie Lens 1M dataset with about 1 million ratings for 3,952 movies by 6,040 users, denoted as ML-1M. In both datasets, ratings vary from 1 to 5. Information on the movies is movie id, movie title, video publishing date, internet address, and 18 genres in which 1 shows that the movie is of that genre and 0 shows that the movie is not of that genre. The movies can also belong to several genres. We use 18 genres as features of a movie in our experiments. For incorporating temporal information in our experiments, the timestamp of ratings was used. Timestamps are represented in seconds in both datasets. For ML-100K and ML-1M, ratings have been issued over 215 days and 1039 days, in the respective order [39]. 3) Netflix Prize dataset includes 100 million ratings issued by 480,000 thousand users for 17,000 movies. The data collection period was between October 1998 and December 2005, reflecting the distribution of obtained ratings. A scale ranging from 1 to 5 was used for ratings [40]. There are no genres in the Netflix Prize dataset. So, we gathered the genres from the IMDB website. Each movie is tagged with at least one genre and a maximum of three genres.

Proposed Algorithm LARC

01: **Input** user u_i, Number of recommendations top_i
02: **Output** List of recommended movies
03: **Assumption**
04: Let *Ai* be learning automaton corresponding to user *ui* with action set *αi*
05: $\alpha_i^j \in \alpha_i$ is associated with a genre $f_i^j \in f_i$
06: *S* denotes the stage number
07: **Begin Algorithm**
08: *S=1*
09: **While** *S<= top*$_i$
10: A_i randomly chooses one of its actions (e.g. α_i^k) at random
11: *W*=Movies that have not been recommended to the user, u_i, and their j^{th} weight feature are 1, are selected.
12: **For** all $m_k \in W$ **D0**
13: Calculate the prediction rate $\hat{r}_{ik}$ with (4)
14: The movie m_k, with the highest, predicted rating, is recommended to the user $u_{i,}$
15: User u_i rates movie m_k with r_{ik}
16: **If** r_{ik}>=*3* **then**
17: **For** all *j* that $f_{m_{j,k}} = 1$ **Do**
18: Reward the actions $\alpha_i^j \in \alpha_i$ (1)
19: **For** all *j* that $f_{m_{j,k}} = 0$ **Do**
20: Penalize the actions $\alpha_i^j \in \alpha_i$ (1)
21: Else
22: **For** all *j* that $f_{m_{j,k}} = 1$ **Do**
23: Penalize the actions $\alpha_i^j \in \alpha_i$ (2)
24: **For** all *j* that $f_{m_{j,k}} = 0$ **Do**
25: Reward the actions $\alpha_i^j \in \alpha_i$ (2)
26: **End IF**
27: S=S+1
28: **End While**

Fig. 2. Proposed approach pseudo

5.3. Recommendation strategy

In this study, the Outcomes of the proposed approach, named LARC, are compared with several existing methods to reveal its superiority. Given our suggested strategy and the available data set, appropriate methods include:

IBGCRS: The First baseline is the algorithm proposed in [27]. The IBGCRS method calculates the association between movie genres according to the rated scores, classifies a movie into a single genre cluster based on the measured correlations. Then, the suggested algorithm determines the genre favored by the target user, categorizes movies belonging to this genre and its comparable genres, and makes a recommendation list involving the identified movies. Finally, the algorithm calculates the listed movies ratings and recommends the intended user accordingly.

CB-Cosin, CB-Jaccard, and CB-Ed: The second baseline is the classical content-based with three schemes CB-Cosine, CB-Jaccard, and CB-Ed that derive from it. These methods recommend items that are similar to the user's preferred items already. We calculate the similarity of movies in terms of genres. These methods use Cosine similarity, Jaccard similarity, and Euclidean Distance similarity as the similarity measure [5].

DTI-CF: DTI-CF method using the time-weighted similarity matrix proposes a dynamic clustering recommender algorithm. This method divides the period into two parts. In each period is processed different network structure and in addition to that, the user's interest signed network model is formed; an enhanced similarity index is used to calculate the users' interest similarity. Then, based on an evolutionary network model, it clustered the users and items. Then, the items are recommended in their cluster according to the collaborative filtering method [10].

GC-MC: GC-MC completes the matrix using a graph auto-encoder framework. The auto-encoder generates the latent features of the item and user nodes using message passing on the bipartite interaction graph. A bilinear decoder reconstructs the rating links applying these latent features. [12].

SCoR: SCoR is a recommender system based on the enhanced version of the Vivaldi synthetic network coordinates system. The item nodes and user nodes are considered replacements for network nodes. Also, distances estimated by user ratings on items are used instead of latencies. System convergence is which means obtaining the desired position for each node. The Euclidean distance between two nodes presents a precise prediction of recommendations [8].

6. Evaluation result

An experimental study on the popular datasets has been done in this section to explore the proposed approach performance. Experiments have been conducted in two categories to appraise the proposed algorithm performance. The first group aims at measuring two metrics of precision, recall, average precision, and average recall, respectively .The second one studies the algorithm accuracy regarding rating prediction.

For these experiments, we calibrated the proposed approach as follows: as the performance of the LARC is dependent significantly on the rewarding rate a and penalizing rate b; so, the right balance can be achieved between the time and quality of the algorithm by selecting the appropriate learning rate. In these tests, learning rates, a and b, range from 0.04 to 0.4. The numerical results demonstrate that by adjusting the reward rate on $a=0.3$ and penalty rate on $b=0.4$, there will be a proper balance between algorithm cost and precision. Coefficients γ and λ are other parameters that must be tuned. These parameters are applied in the proposed rating prediction formula. The numerical results confirm that in these experiments, $\gamma=0.4$ and $\lambda=0.6$ lead to the best accuracy.

In this paper, ratings are indexed based on the day. First, we sorted the dataset for each user by time using the timestamp in the experimental data. Then for each user, we divided experimental datasets into training and a test set. The training set contains 80% of the beginning of sorted data of each user, and the testing set is the remaining 20%. We performed the experiments 20 times. In the following, the average evaluation results for the test set are presented. All experiments are executed on a PC with Intel processor Core i5 @ 1.8 GHz and 4 GB RAM applying MATLAB software (version 2017) running on the Windows platform.

6.1. Evaluation of quality recommendations

These experiments evaluated the quality of the recommendation list of algorithms. We compared the LARC model with IBGCRS, CB-Cosin, CB-Jaccard, CB-Ed, and DTI-CF.

To investigate the effect of the length of the recommendation list, N, on the performance of approaches, we compare the quality of recommendation lists with different sizes ranging from 1 to 10 at intervals of 1. Figs. 3-10 show the results.

Figs. 3 and 4 depict the precision of LARC in comparison with IBGCRS, CB-Cosin, CB-Jaccard, and CB-Ed on datasets ML-1M and Netflix Prize. As expected, the precision of investigated methods is almost constant when changing the length of the recommendation list; because these methods recommend movies to the user that are related to him. Notice that the LARC outperforms IBGCRS and three content-based algorithms since it recommends movies based on the users' interests. LARC tracks the change of users' interests by learning automata over time. The learning automaton is capable of computing the user's interest in each genre through user feedback. The user feedback is user rating to a recommended movie in user interaction with the recommender system. The actions probability vector of learning automata is updated based on this feedback. Therefore, the actions probability vector of learning automata converges to the user's degree of interest for each genre, resulting in increased precision. IBGCRS outperforms content-based methods in terms of precision. The content-based methods precision scores are close and the worst because they are just different in the similarity measure.

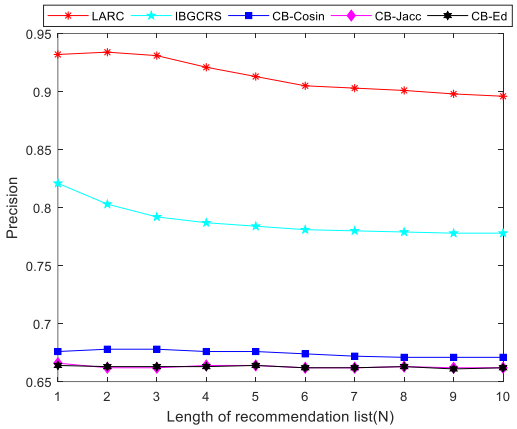


Fig. 3. Precision for the different algorithms with the various length of recommendation list on ML-1M.

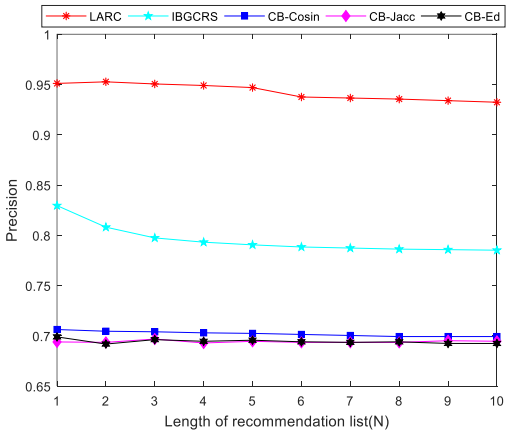


Fig. 4. Precision for the different algorithms with the various length of recommendation list on Netflix Prize.

Fig. 5 demonstrates the precision of LARC in comparison with mentioned algorithms and the DTI-CF method on dataset ML-100K. As shown, the precision of investigated methods on dataset ML-100K is less flexible in changing the length of the recommendation list. LARC is more robust in terms of precision than the other algorithms because a learning automaton tracks the change of user's interests by applying the user rating as described in the previous paragraph. The DTI-CF method performs better than IBGCRS, CB-Cosin, CB-Jaccard, and CB-Ed.

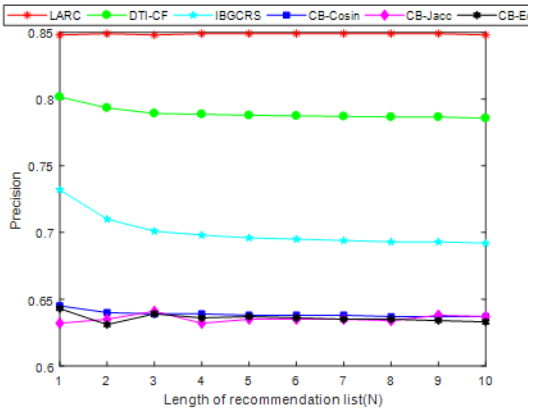


Fig. 5. Precision for the different algorithms with the various length of recommendation list on ML-100K.

The recall value is greater than the precision value because the number of recommended movies is lower than the existing movies in the dataset. We examined the impact of length of the recommendation list on recall of investigated methods on datasets ML-100K, ML-1M, and Netflix Prize. Figs. 6-8 depict the results of our experiments. As predicted, with the increasing length of the recommendation list, recall increases since more favorite movies are recommended.

According to Figs. 6 and 7, the LARC method gets the best recall, followed by IBGCRS. The recalls of the three other methods are almost equal, and their recall is the worst. Fig.8 demonstrates better recall values for the LARC compared to other methods on the ML-100K dataset. Among the different methods, the DTI-CF method outperforms the traditional methods of content-based methods and IBGCRS. The content-based methods perform the worst among all the methods.

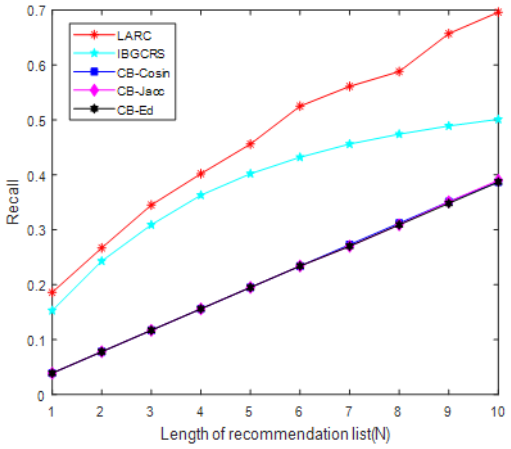


Fig. 6. Recall for the different algorithms with the various length of recommendation list on ML-1M.

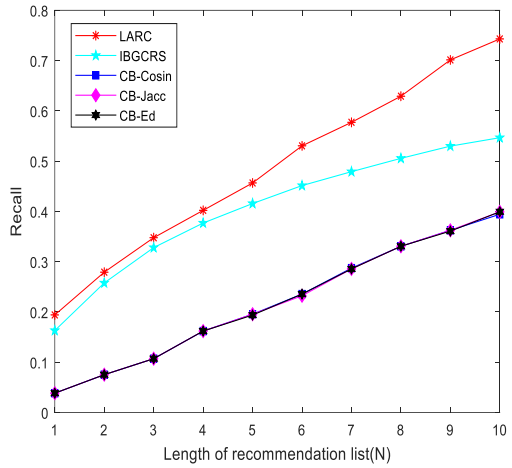


Fig. 7. Recall for the different algorithms with the various length of recommendation list on Netflix Prize.

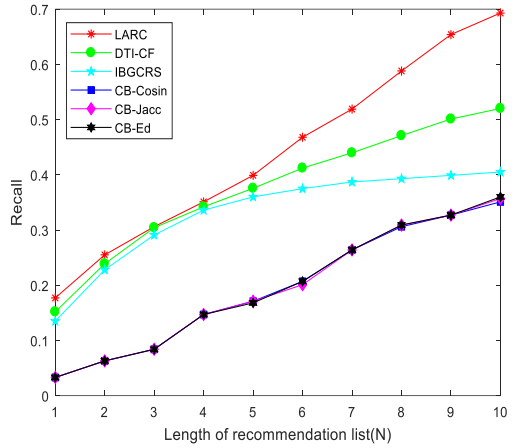
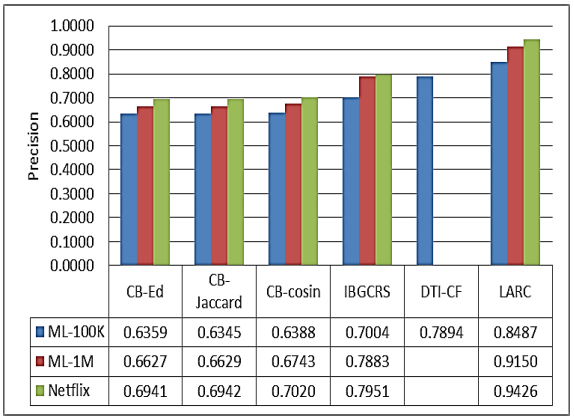


Fig. 8. Recall for the different algorithms with the various length of recommendation list on ML-100k.

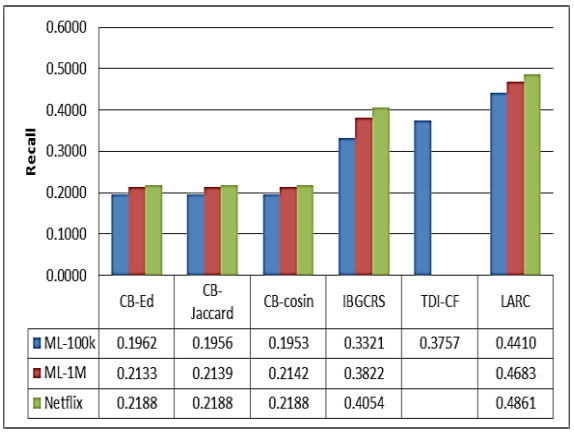
Another experiment is done for determining the quality of algorithms recommendations list based on the average precision and average recall. Figs 9 and 10 show average precision and an average recall of the proposed method (LARC) compared TDI-CF, IBGCRS, CB-Cosin, CB-Jaccard, and CB-Ed on datasets ML-100K and ML-1M, and Netflix Prize, respectively. From the results given in Figs 9 and 10, it is evident that LARC significantly outperforms the other methods in terms of average precision and average recall; because learning automaton adapts itself to changing user interests. For example, average precision values on three datasets ML-100K, ML-1M, and Netflix; LARC excels IBGCRS 21.1%, 16%, and 18.5% respectively; LARC outperforms IBGCRS 32.7%, 19.5%, and 19.9% in terms of average recall.

Besides, among the different methods, conducting the DTI-CF approach on the ML-100K dataset outperforms the traditional content-based methods and IBGCRS. DTI-CF excels IBGCRS and the derived approaches of content-based filtering 12.7% and 24% in terms of average precision; also outperforms 13.1% and 99.8% in the average recall measure. The results also reveal that IBGCRS outdoes CB-Cosin, CB-Jaccard, and CB-Ed; thus, they are classified below DTI-CF. A comparison between the results in Figs 9 and 10 shows the movies recommended by classical content-based methods provide the least satisfaction to the user; also, the precision and recall of CB-Cosin, CB-Jaccard, and CB-Ed are equal. One reason for this equality is that these differ only in calculating the similarity of movies. In addition, results in two Figs 9 and 10 show that the size of the datasets influenced the quality of the recommendation list of algorithms. The quality of the recommendation list of all algorithms conducted on the Netflix Prize dataset is the best.



	CB-Ed	CB-Jaccard	CB-cosin	IBGCRS	DTI-CF	LARC
ML-100K	0.6359	0.6345	0.6388	0.7004	0.7894	0.8487
ML-1M	0.6627	0.6629	0.6743	0.7883		0.9150
Netflix	0.6941	0.6942	0.7020	0.7951		0.9426

Fig. 9. Average precision for different algorithms on ML-100k, ML-1M, and Netflix.



	CB-Ed	CB-Jaccard	CB-cosin	IBGCRS	TDI-CF	LARC
ML-100k	0.1962	0.1956	0.1953	0.3321	0.3757	0.4410
ML-1M	0.2133	0.2139	0.2142	0.3822		0.4683
Netflix	0.2188	0.2188	0.2188	0.4054		0.4861

Fig. 10. Average recall for different algorithms on ML-100k, ML-1M, and Netflix.

6.2. Evaluation of accuracy predicted ratings

The third part of the experiments was focused on measuring the accuracy of LARC rating

prediction compared to the IBGCRS, DTI-CF, GC-MC, and SCoR algorithms in some terms RMSE and MAE. This experiment tested algorithms on datasets ML-100k, ML-1M, and Netflix Prize. We summarized the obtained results in Table 1. Comparison of the results shows that predicting the ratings in the LARC algorithm using equation (4) is more accurate than predicting the ratings obtained from the IBGCRS and DTI-CF algorithms. To predict the rating, equation (4) uses coefficients of two values. The first value is the average rating given on the candidate movie by other users; the second value is the average rating of the active user on movies whose features are similar to features of the candidate movie. In against, the IBGCRS and DTI-CF algorithms are applying the collaborative filtering method only. The IBGCRS method defines the average ratings of other users in the movie as a rating prediction. For example, in prediction performance LARC outperforms IBGCRS 8.3%, 5.7%, and 5.6% in RMSE, as well as 24.1%, 14.8%, and12.7% in MAE on ML-100K, ML-1M, and Netflix respectively. In the DTI-CF method, the prediction ratings are calculated based on the collaborative filtering in each cluster. Fewer prediction errors lead to higher accuracy of recommendations. According to Table 1, the accuracy of the GC-MC method in term of RMSE on ML-100K and Ml-1M datasets are better than our LARC. The rating prediction of the LARC algorithm compared to SCoR in terms of RMSE on the ML-100K dataset is less precise. Specifically, SCoR outperforms LARC 15.2% in RMSE on ML-100K. GC-MC outperforms 10.7% and 20.7% in RMSE on ML-100K and ML-1M respectively.

Table.1.
RMSE and MAE

	ML-100K		***ML-1M***		***Netflix***	
Algorithm	RMSE	MAE	RMSE	MAE	RMSE	MAE
LARC	1.008	0.618	1.005	0.634	0.968	0.574
IBGCRS	1.092	0.767	1.063	0.728	1.023	0.647
DTI-CF	1.010	0.721	—	—	—	—
GC-MC	0.910	—	0.832	—	—	—
SCoR	0.875	—	—	—	—	—

6.3. Discussion

The user's preferences are dynamic and temporal, and they change over time. User ratings reflect user interests. Hence, analyzing the user ratings shows promise of performance improvement. User rating to each movie shows his interest in the movie genres. Therefore, we proposed LARC to learn the degree of user interest in the genres personally. This method tracks the changes in user interests during the time through the interactions between the user and system. The results in the previous sections demonstrate that awareness of user interests is surprisingly really essential for recommending.

A drawback of this approach is using online user feedback (user ratings) that in every three data sets ML-1K, ML-1M, and Netflix are 5-point ratings. While using these datasets enables comparative studies, but it limits the generalizations of results to other domains. The star rating (online feedback) is a usual solution to outline the overall rate of the review briefly. The rating request may be missed or ignored by many users. On the other hand, the system request to provide feedback may be upset and disruptive to the recommendation process, probably influencing the user behavior. Besides, the ratings provided by the users may not be the ideal index of their received service quality and satisfaction; also, they may be manipulated either by the business proprietors or adversaries; therefore, ratings of users inherently have noise[3].

7. Conclusions and suggestions

The present study proposed a personalized recommendation system based on learning automata to track the changes in user interests. In this method, learning automata is responsible for learning user interests, where these Interests are actions of learning automata. The user rating is a response of the environment to the learning automata. The user's interests are formed gradually, relying on user ratings over time through the interactions between the user and learning automata, leading to enhanced quality and accuracy of the recommendation list. Compared with the previous methods, we achieved a more genuine and accurate interests vector. In addition, we modeled the user's interest according to the features of movies making the modeling granularity more sensible. Three well-known databases, namely ML-100K, ML-1M, and Netflix Prize, are employed to evaluate our approach and compare it with IBGCRS, SCoR, DTI-CF, GC-MC, and classical content-based algorithms in terms of precision, recall, and accuracy.

In brief, LARC can gradually improve performance according to the user feedbacks online impressively. Consequently, this method was adjusted to various environments with different user requirements. Thus, the proposed algorithm fits into the personalization of the recommender systems. In addition, we believe multi-level automata have a tremendous potential to extend the LARC since they can apply multi-features of items to model the user profile on different levels.

References

[1] Adomavicius, G. and A. Tuzhilin, Personalization technologies: a process-oriented perspective. Communications of the ACM, 2005. 48(10): p. 83-90.

[2] Gauch, S., et al., User profiles for personalized information access, in The adaptive web. 2007, Springer. p. 54-89.

[3] Jawaheer, G., P. Weller, and P. Kostkova, Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. ACM Transactions on Interactive Intelligent Systems (TiiS), 2014. 4(2): p. 8.

[4] Cufoglu, A., User profiling-a short review. International Journal of Computer Applications, 2014. 108(3).

[5] Bobadilla, J., et al., Recommender systems survey. Knowledge-based systems, 2013. 46: p. 109-132.

[6] Deshpande, M. and G. Karypis, Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS), 2004. 22(1): p. 143-177.

[7] Staab, S. and R. Studer, Handbook on ontologies. 2010: Springer Science & Business Media.

[8] Papadakis, H., C. Panagiotakis, and P. Fragopoulou, SCoR: a synthetic coordinate based recommender system. Expert Systems with Applications, 2017. 79: p. 8-19.

[9] Chen, J., H. Wang, and Z. Yan, Evolutionary heterogeneous clustering for rating prediction based on user collaborative filtering. Swarm and Evolutionary Computation, 2018. 38: p. 35-41.

[10] Chen, J., et al., A Temporal Recommendation Mechanism Based on Signed Network of User Interest Changes. IEEE Systems Journal, 2019.

[11] Tran, T., et al. Regularizing matrix factorization with user and item embeddings for recommendation. in Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018.

[12] Berg, R.v.d., T.N. Kipf, and M. Welling, Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263, 2017.

[13] Wang, X., et al. Neural graph collaborative filtering. in Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. 2019.

[14] Lops, P., M. De Gemmis, and G. Semeraro, Content-based recommender systems: State of the art and trends, in Recommender systems handbook. 2011, Springer. p. 73-105.

[15] Pazzani, M.J. and D. Billsus, Content-based recommendation systems, in The adaptive web. 2007, Springer. p. 325-341.

[16] Van Meteren, R. and M. Van Someren. Using content-based filtering for recommendation. in Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop. 2000.

[17] Burke, R., Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 2002. 12(4): p. 331-370.

[18] Cremonesi, P., R. Turrin, and F. Airoldi. Hybrid algorithms for recommending new items. in Proceedings of the 2nd international workshop on information heterogeneity and fusion in recommender systems. 2011. ACM.

[19] Jo, S., Weight Recommendation Technique Based on Item Quality To Improve Performance of New User Recommendation and Recommendation on The Web. Hannam University Graduation School: Doctoral thesis, 2008.

[20] Symeonidis, P., A. Nanopoulos, and Y. Manolopoulos. Feature-weighted user model for recommender systems. in International Conference on User Modeling. 2007. Springer.

[21] Zhang, J., et al., Collaborative filtering recommendation algorithm based on user preference derived from item domain features. Physica A: Statistical Mechanics and its Applications, 2014. 396: p. 66-76.

[22] Ticha, S.B., et al. User-feature model for hybrid recommender system. in 4th International Conference on Information Systems and Economic Intelligence-SIIE'2011. 2011. IGA Maroc.

[23] Gou, Z., et al., Personalized Search by a Multi-type and Multi-level User Profile in Folksonomy. Arabian Journal for Science and Engineering, 2018: p. 1-10.

[24] Veloso, B., et al., Improving On-line Genre-based Viewer Profiling. 2017.

[25] Kassak, O., M. Kompan, and M. Bielikova, User preference modeling by global and individual weights for personalized recommendation. Acta Polytechnica Hungarica, 2015. 12(8): p. 27-41.

[26] Li, J., et al., Movie recommendation based on bridging movie feature and user interest. Journal of computational science, 2018. 26: p. 128-134.

[27] Hwang, T.-G., et al., An algorithm for movie classification and recommendation using genre correlation. Multimedia Tools and Applications, 2016. 75(20): p. 12843-12858.

[28] Hu, Y., et al. A hybrid genre-based personalized recommendation algorithm. in Industrial Electronics and Applications (ICIEA), 2016 IEEE 11th Conference on. 2016. IEEE.

[29] Frémal, S. and F. Lecron, Weighting strategies for a recommender system using item clustering based on genres. Expert Systems with Applications, 2017. 77: p. 105-113.

[30] Zhang, C., et al. Are Features Equally Representative? A Feature-Centric Recommendation. in AAAI. 2015.

[31] Phuong, N.D. and T.M. Phuong. A graph-based method for combining collaborative and content-based filtering. in Pacific Rim International Conference on Artificial Intelligence. 2008. Springer.

[32] Forsati, R. and M.R. Meybodi, Effective page recommendation algorithms based on distributed learning automata and weighted association rules. Expert Systems with Applications, 2010. 37(2): p. 1316-1330.

[33] Talabeigi, M., R. Forsati, and M.R. Meybodi. A hybrid web recommender system based on cellular learning automata. in Granular Computing (GrC), 2010 IEEE International Conference on. 2010. IEEE.

[34] Ghavipour, M. and M.R. Meybodi, An adaptive fuzzy recommender system based on learning automata. Electronic Commerce Research and Applications, 2016. 20: p. 105-115.

[35] Lakshmivarahan, S. and M. Thathachar, Bounds on the convergence probabilities of learning automata. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 1976. 6(11): p. 756-763.

[36] Narendra, K.S. and M.A. Thathachar, Learning automata-a survey. IEEE Transactions on systems, man, and cybernetics, 1974(4): p. 323-334.

[37] Billard, E. and S. Lakshmivarahan, Learning in multilevel games with incomplete information. I. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1999. 29(3): p. 329-339.

[38] Narendra, K.S. and M.A. Thathachar, Learning automata: an introduction. 2012: Courier Corporation.

[39] https://grouplens.org/datasets/movielens/.

[40] https://ia800205.us.archive.org/7/items/nf_prize_dataset.tar/nf_prize_dataset.tar.gz.