Like-tasted user groups to predict ratings in recommender systems

Soufiene Jaffali, Salma Jamoussi, Kamel Smaïli, Abdelmajid Ben Hamadou

To cite this version:

Soufiene Jaffali, Salma Jamoussi, Kamel Smaïli, Abdelmajid Ben Hamadou. Like-tasted user groups to predict ratings in recommender systems. Social Network Analysis and Mining, Springer, 2020, 10 (1), 10.1007/s13278-020-00643-w . hal-02867310

HAL Id: hal-02867310

https://hal.archives-ouvertes.fr/hal-02867310

Submitted on 14 Jun 2020

Noname manuscript No.
(will be inserted by the editor)

Like-Tasted User Groups to Predict Ratings in Recommender Systems

Soufiene Jaffali · Salma Jamoussi · Kamel Smaili · Abdelmajid Ben Hamadou

the date of receipt and acceptance should be inserted later

Abstract Recommendation Systems have gained the intention of many researchers due to the growth of the business of personalizing, sorting and suggesting products to customers. Most of rating prediction in recommendation systems are based on customer preferences or on the historical behavior of similar customers. The similarity between customers is generally measured by the number of times customers liked or disliked the same item. Given the huge number and the variety of items, many customers cannot be considered as similar, as they did not evaluate the same items, even if they have similar tastes. This paper presents a new method of rating prediction in recommendation systems. The proposed method starts by identifying the taste directions or the interest centers based on the users' demographic information combined with their previous evaluations. Thus, it uses the Principal Component Analysis (PCA) to retrieve the major taste orientations. According to these orientations, user groups are created. Then, for each group, it generates a prediction model, that will be used to predict unknown rates of users within the corresponding group. In order to assess the accuracy of the proposed method, we compare its results with four baseline methods, namely: RegSVD, BiasedMF, SVD++ and MudRecS. Results prove that the proposed algorithm is more accurate than the baseline algorithms.

Keywords Rating prediction · Movielens · PCA · Like-Tatsed users

S. Jaffali
Community College,
Imam Abdurrahman bin Faisal University,
Dammam, KSA
E-mail: smjaffali@iau.edu.sa

S. Jamoussi and A. Ben Hamadou
University of Sfax,
Sfax 3029, Tunisia
E-mail: {jamoussi, abdelmajid.benhamadou}@gmail.com

K. Smaili
SMART, LORIA,
Vandoeuvre les Nancy, 54506, France
E-mail: smaili@loria.fr

1 Introduction

The Recommendation System (RS) consists of suggesting items and offering advice and directions for customers that improve decision making process. Moreover, RS decreases transaction costs of selecting products and increases e-commerce earnings. The suggestions must take into account the customer's profile and his previous choices in a manner that the suggestions suit to the customer's tastes.

In this context, several recommendation methods and algorithms are proposed. The majority of these algorithms use the matrix $R : users \times items$, containing stored evaluations [3]. Given that the ratings are not available for all tuples $(user, item)$, the matrix R suffers from data sparseness (the majority of the values are missing) [12]. In order to predict the missing ratings, the proposed algorithms are based on the partial matrix factorization [12]. Given the huge number of items to recommend and the number of users on one hand, and the small number of known $user - item$ ratings on the other hand, the problem with these solutions is the data sparseness. Moreover, these algorithms are unable to predict recommendations for a new item, which is not in the training data, or a new user.

The main contribution of this work is presenting a new method to recommend items based on like-tasted user groups. The proposed method aims at utilizing the recorded ratings to represent the users' interest centers (tastes of users). Then, users are grouped according to their tastes. To predict an unrecorded rating for a tuple $(user, item)$, the history of the user's ratings, as well as the recorded ratings of other users belonging to his group, are used. Such a method improves the quality of recommendations and decreases the complexity of the prediction task. Indeed, to create a movie recommendation model for a homogeneous group of users, i.e. having similar cinematographic interests, is easier.

In this paper, a new method of recommendation is presented. This method is called *GLER (Grouping Like-tasted users to Estimate Ratings)* and aims at predicting rates based on the historical ratings of like-tasted users. The remaining of this paper is organized as follows. In Section 2, we present some related works on RS. Section 3 presents the research objectives. Section 4 describes the proposed method. We discuss the obtained results in Section 5. Finally, we conclude and present future directions in Section 6.

2 Related work

Collaborative Filtering (CF) techniques are considered to be the most popular RS approaches [20]. These techniques recommend items to a user u according to what other users, with similar interests, have liked previously. Two users are considered as similar if they have similar rating history. Koren [17] distinguishes two main approaches of CF; namely the *neighborhood models* and the *latent factor models*. In order to estimate ratings, earlier neighborhood models are based on a user-user methods. The essential idea behind these methods is to exploit the stored ratings of users having similar tastes. Several functions have been suggested to evaluate the similarity between users, such as the Pearson correlations and the cosine similarity function.

The *user-user* methods create E a set of n users similar to the user u. Then combine the evaluations of the users within the set E to predict preferences for the user u towards an item i. The prediction is usually done by calculating the weighted average of the ratings assigned by similar users for the item i based on the following formula:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in E} s(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in E} |s(u,v)|} \tag{1}$$

Where E denotes the set of users similar to u and $s(u,v)$ is the similarity between the user u and the user v.

In addition to the user-user methods, new methods known as item-item methods became popular. The main idea of these methods is to estimate the ratings of a user u on a given item i based on the recorded ratings made by u on similar items [18]. The cosine function is usually used to calculate the similarity between two items i and j. After calculating the similarity between items, item-item methods collect a set S containing k items similar to the item in question. The evaluation of the user u for the item i can be predicted as follows:

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{2}$$

Where the $p_{u,i}$ is equal to the sum of ratings that the user u has given to items in S. Each rating $r_{u,j}$ is weighed by the similarity between the item i and the item j. This sum is normalized by the sum of similarities between the item i and all the items in S.

The huge number of items to recommend and the long tail distribution of rating frequencies [5] are the main problems of the neighborhood-based approach. Indeed, focusing on the neighborhood ratings and considering only items rated by similar users leads to ignoring a considerable part of items and limits the coverage of recommendations. This later can also be resulted from the fact that the similarity of two users is given by comparing their ratings for the same items, and users cannot be similar if they did not rate exactly the same items, even if those items are similar. Furthermore, as the majority of users assess only a small proportion of the available items [22], sparsity is a common issue for most recommendation systems [12].

Some techniques based on latent factor models are proposed to overcome coverage and sparsity problems. Latent factor models techniques represent users and items in a compact and meaningful way that reflects their most significant features [22]. The essential thought behind latent factor models is to build a prediction model based on the stored ratings. Once built, the model is used to estimate unknown ratings [14]. Many latent factor models techniques are utilized in the context of CF, such as *Matrix Factorization* [17], *Probabilistic Matrix Factorization* [21] and other variations [30]. In this context, the *SVD* technique is deeply used in CF due to its accuracy and scalability [10, 23]. Recent recommendation tools like *LingPipe* and *pyrsvd* are mostly based on models derived from *SVD* such as *SVD++* and *timeSVD++* [1, 17]. A more detailed presentation of some variants of the *SVD* technique is in section 5.3.

Despite being a well-known matrix factorization method, to our knowledge, only the *Eigentaste* algorithm [9] uses *Principal Component Analysis (PCA)* in CF for joke recommendation. *Eigentaste* establishes $n \times m$ matrix denoted $\hat{R}$ ($\hat{r}_{u,i} = (r_{u,i} - \mu_i)/\sigma_i$), representing the stored evaluations of n users for m items. Where μ_i is the mean rating given to the item i and σ_i is the standard deviation of the ratings given to i. In the second step, the PCA calculates the correlation matrix

$C = \frac{1}{|U|-1}\hat{R}^T\hat{R}$ and a factorization $C = E^T \Lambda E$, where E is an orthogonal matrix of eigenvectors of C and Λ is a diagonal matrix containing the eigenvalues of C. The k best eigenvalues are retained, and the resulting factorization projects the users into a k-dimensional space. Finally, *Eigentaste* groups users into the space of dimension k (with $k = 2$) by Recursive Rectangular Clustering, and recommends jokes to a user u based on the preferences of other users in the same group.

To deal with the sparsity and scalability problems, that CF techniques suffer from, some works propose a RS based on data clustering. In this context, Das et al. [6] use DBSCAN clustering algorithm for clustering users according to their preferences. To recommend items of interest for a new user, authors use different voting systems as algorithms to combine opinions from multiple users within his cluster. Alam et al. [2] generate recommending patterns using Hierarchical Particle Swarm Optimization based clustering (HPSO-clustering). In [29], the authors present a k-means clustering-based recommendation algorithm, in order to address the RS scalability issues.

3 Research objective

Based on the literature review presented in the previous section, this work aims at proposing a new method that improves rating prediction accuracy by addressing the sparseness and limited coverage problems. The following points present the novelty of this work:

- In contrast to most rating prediction algorithms that record only ratings for tuples $(users, items)$, we represent each user by his demographic information. Thus, we can predict ratings for a new user (without rating history) based on recorded ratings of users with closest demographic features. In addition to the demographic information, we add the recorded ratings for item features to the user's representation. In this manner, we are able to predict ratings for new items and items in the long tail based on their features.
- This method uses PCA to retrieve the major interest centers according to item features. Next, it projects the data into the space of interest centers. These interest centers are the common tastes shared by users. Then, we group users according to their tastes. Finally, we build a prediction model for each like-tasted user group. Thus, the rating for a tuple (u, i) is predicted according to previous ratings given by the user u and all users within his group. In this way, we enrich the user u rating history by ratings of other users having similar tastes. Furthermore, building a prediction model for a reduced number of users having similar tastes is less complex and generates more accurate results.
- The use of PCA leads to decreasing the impact of data sparseness. Indeed, PCA is widely used to reduce the noise and the dimensions of data [13].

4 Proposed method

The main idea of the *GLER* method is to build like-taste user's groups according to their interest centers or tastes. These groups are used to predict unknown ratings of a tuple *(user u , item i)* based on previous ratings of the user u and other users within his group.

In order to predict the users interests towards items using *GLER*, we start by creating a matrix M containing the recorded evaluations. Then, common interest centers are sought, using the *Principal Component Analysis (PCA)* technique. Next, the matrix M is projected into the space of interest centers, and users are grouped according to those centers. Finally, a regression algorithm is utilized to create a prediction model for each user group. These models are used to predict the future ratings of each pair (user u, item i) based on the history of ratings recorded by the users within the group to which u belongs. Figure 1 illustrates the different steps of *GLER*.

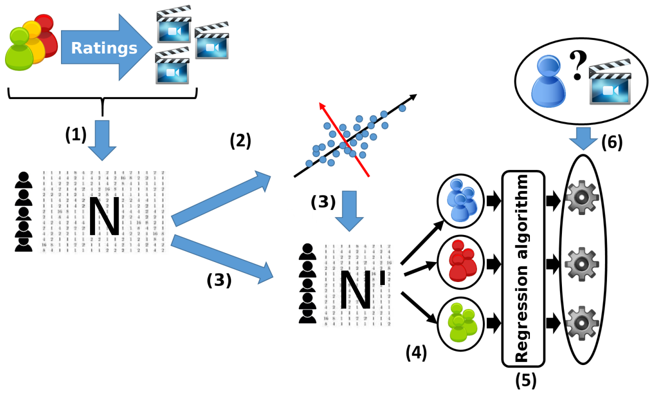

Fig. 1 The proposed method steps.

The rest of this section describes the steps of the proposed method. These steps can be organized in two main phases: *(i)* the grouping like-tasted users phase, and *(ii)* the rating prediction phase.

4.1 Grouping like-tasted users

4.1.1 Interest centers matrix

In the first phase of *GLER*, the users are grouped together according to their interest centers. To do this, we start by creating a matrix M representing the users interests. This matrix consists of n row vectors. Each row vector corresponds to a user u and contains the means of the evaluations that u attributed by genre or category of item (such as action, comedy and drama for movies). To these information, a demographic information such as age and gender is added to enrich the users representation.

Once the M matrix is made, we produce the matrix N by normalizing the M matrix values. Therefore, each value in M is divided by the standard deviation σ of each column:

$$N_{ui} = \frac{M_{ui}}{\sigma_i} \tag{3}$$

4.1.2 Retrieving interest centers

We apply *PCA* to retrieve latent factors representing users interest. Thus, we begin by computing the covariance matrix of N, denoted C.

$$C = \frac{1}{n-1} N^T N \tag{4}$$

Then, the matrices Σ and Z are retrieved, by solving the equations (5) and (6):

$$C = Z^T \Sigma Z \tag{5}$$

$$ZCZ^T = \Sigma \tag{6}$$

Where Σ is a diagonal matrix. Elements of Σ are the eigenvalues of the matrix C. The columns of Z are the eigenvectors of C. The eigenvectors of C represent the latent interest centers, and define in $\mathbb{R}^n$ the orientations of the principal components (interest centers) of the data. The eigenvalues represent the importance of each of these centers. They correspond to the variances of data when projected on each of these components.

Given the fact that the eigenvectors represent the latent interest centers, to know the number of eigenvectors maximizing the covariance and expressing the majority of information is to know the number of interest centers, and so the number of groups to consider. The eigenvalues in Σ are used to identify v, the number of eigenvectors representing the majority of information. Therefore, we apply the STAF method (Scree Test Acceleration Factor) [25], which consists in retaining the eigenvectors corresponding to the eigenvalues preceding the coordinate where the acceleration factor af is maximized. The acceleration factor is indicated by the sudden change in the slope of the curve of f. With, f is the function passing through all the eigenvalues.

After calculating the number of groups to consider, the data are projected into the interest centers space formed by the retained eigenvectors. The result is the following matrix N':

$$N' = N Z_v^T \tag{7}$$

With Z_v is the matrix formed by the retained eigenvectors.

The obtained matrix N' represents the users new coordinates in the space of the interest centers. In this space, users with similar tastes (interest centers) will have close coordinates, and conversely, users with divergent orientations will have remote coordinates.

4.1.3 Creating user groups

We adopt the *K-means* algorithm to build user groups. This choice is argued by the efficiency and simplicity of *K-means*. As input, *K-means* takes the matrix N' and the number of user groups v retrieved in the previous step.

The matrix N' is the projection of the input data in the space of interest centers. In this representation, users who have similar tastes will have close coordinates in the new space. This representation is more suitable to the grouping step using the *K-means* algorithm, which seeks to bring together the closer users. Thus, users with similar cinematographic tastes and who have close coordinates in the space of interest centers, have a high probability of being assigned to the same group.

4.2 Predict ratings

For each group of users created, we utilize the recorded ratings given by users within this group as the training set for the prediction algorithm. Once learned, the system is subsequently able to predict the interests of users within the group. To predict interests of a new user (who does not figure in the input data), he will be assigned to the nearest group depending on the cosine distance between its representative vector and the group representative vector. Each group representative vector is the average of the user's vectors within the group.

Rating prediction can be considered either as a regression problem or as a classification problem. If the evaluations are in the form of classes or labels (*I like / I do not like*, *good / bad*), the prediction is considered as a classification problem. On the other hand, the prediction is considered as a regression problem, if the evaluations are given in the form of numerical values (such as the evaluation degrees from 1 to 5 in the *MovieLens* Recommendation System).

In this work, we deal with movies rating prediction (Section 5). The ratings form a range of discrete values from 1 to 5, where the rating level 2 is closer to the rating level 1 than the rating level 5, and do not present five different classes. In this case, RSs seek to approximate the rating value to predict. Therefore, to estimate the movie's rating, we use a regression algorithm instead of a classification algorithm.

In the literature, many regression algorithms have been applied. In this work, three regression algorithms, widely used, have been adopted, namely: the *M5P* algorithm, the *Multi-Layer Perceptron* and the *Support Vector Regression.*

4.2.1 Regression trees (M5P)

This algorithm is proposed to induce regression model trees. It selects attributes minimizing the expected error to construct regression trees nodes. The node leaves in the constructed regression trees are composed of multivariate linear models. This algorithm has proved its efficiency in several predicting applications, such as, the case of predicting missing values in brains suffering from a traumatic lesion [7], and the case of the recipes recommendation [8].

4.2.2 MultiLayer Perceptron (MLP)

The *Multi-Layer Perceptron* is a multi-layered neural network classifier, that can be considered as a logistic regression classifier where the input is transformed using a given non-linear transformation Φ. This transformation projects the input data into a space where they become linearly separable.

4.2.3 Support Vector Regression (SVR)

Is the regression form of *Support Vector Machines (SVM)*. *SVR* is based on the principle of *SVM*, except that *SVR* introduce an alternative loss function including a distance measurement. In this work we adopt two well-known alternatives to *SVR*, namely: ν-*SVR* [26] and ε-*SVR* [27].

In ν-*SVR*, the parameter ν is used to identify the proportion of the number of support vectors to keep with respect to the total number of samples in the dataset. In ε-*SVR* there is no control on how many data vectors from the dataset becomes support vectors. Yet, ε-*SVR* gives a total control of how much error the regression model allowed to, and anything beyond the specified ε will be penalized in proportion to the regularization parameter.

5 Experimentations

In the experimentation, the *GLER* method is applied to predict ratings in the context of movies' recommendation which is a well known field in the RS domain. In this area, several evaluation environments have been proposed, including *MovieLens* and *Netflix.* The challenge of *Netflix* is an open competition for the best collaborative filtering algorithm to predict user ratings for movies, based on recorded ratings. *MovieLens* is a recommendation system and a virtual community website. Based on collaborative filtering, *MovieLens* recommends movies for its customers, using their cinematographic preferences. *MovieLens* and *Netflix* tasks have attracted several researchers, and many algorithms have been proposed for this purpose.

In this section, we start by presenting the dataset, the evaluation metrics, baseline methods and the experimentation protocol. Then, we present and discuss the obtained results.

5.1 DataSet

The *MovieLens-100K* dataset is widely used to assess recommender systems efficiency [4, 11]. It was collected as a part of the *GroupLens* research project, through the *MovieLens* website. This dataset contains 100,000 evaluations in the form of ratings between 1 and 5, given by 943 users for 1682 movies. It contains also demographic information about users such as *age*, *gender*, *occupation* and *zip-code.* For each movie, the dataset specifies the *title*, the *release date*, the *video release date*, its *IMDB URL* and its *cinematographic genre* (action, horror, comedy, drama, etc.). A movie may correspond to one or multiple genres.

The data is distributed over five sets of data. Each set, is split into a training set and a test set composed of respectively 80,000 and 20,000 entries. These entries are given in the following format:

User ID — Movie ID — Rating — Timestamp

5.2 Evaluation Metrics

In the recommendation system frameworks, typically, the data is divided into a training set R_{learn} and a test set R_{test}. The training set is utilized to generate models and adjust the recommendation system settings. The

R_{test} set is used to evaluate the recommendation system. Let $\hat{r}_{ui}$ the ratings predicted by the system, and r_{ui} the recorded ratings. RSs aim to estimate ratings in R_{test} based on the prediction models created according to the recorded ratings in the training set. The closer $\hat{r}_{ui}$ and r_{ui} are, the more accurate the recommendation system is.

In the literature, several metrics are used to evaluate recommendation systems according to the studied property (prediction accuracy, scalability, diversity, adaptivity) [15]. To evaluate the predictions accuracy, we use two commonly used error measures: the *Root Mean Square Error* and the *Mean Absolute Error*, that are explained in the remainder of this paragraph.

5.2.1 Root Mean Square Error (RMSE)

The *Root Mean Square Error* between the predicted and recorded ratings is provided by:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in R_{test}} (\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}} \tag{8}$$

Where $|R_{test}|$ is the cardinality of the test set R_{test}.

5.2.2 Mean Absolute Error (MAE)

The *Mean Absolute Error* is a very known alternative of *RMSE*. It is given by the following equation:

$$MAE = \frac{\sum_{(u,i)\in R_{test}} (\hat{r}_{ui} - r_{ui})}{|R_{test}|} \tag{9}$$

The difference between the *RMSE* and the *MAE* lies in the fact that the first prefers systems that make low errors, while the second favors systems making less number of errors. Table 1 illustrates the following example: given two recommendation systems A and B, and five rating values to predict. The system A makes an error of 1 on four ratings and an error of 0 on the fifth rating. The system B makes an error of 4 on one rating and error of 0 on the other ratings. The *RMSE* prefers the first system and penalizes the second contrary to the *MAE*.

Table 1 illustrative example of the evaluation using RMSE and MAE.

R	$\hat{R}_{\text{system-A}}$	$\hat{R}_{\text{system-B}}$
5	4	1
4	3	4
4	5	4
3	2	3
2	2	2
RMSE	$\sqrt{\frac{4}{5}}$	$\sqrt{\frac{9}{5}}$
MAE	$\frac{4}{5}$	$\frac{3}{5}$

5.3 Baseline

In this paragraph, we present four well known rating prediction methods, namely *RegSVD*, *BiaisedMF*, *SVD++* and *MudRecS*. These methods will be used as baselines to evaluate and position our method.

5.3.1 RegSVD

The *Regularized Singular Value Decomposition (RegSVD)* is based, on one hand, on the *Singular Value Decomposition* and, on the other hand, on the hypothesis that h unknown factors can be utilized to approximate a given matrix.

RegSVD calculates the rating that a user u_i will give to a movie v_j by introducing two vectors p_i and q_j of dimensions h. p_i represents the tastes vector of the user u_i. This vector illustrates how this user is interested in the selected factors. q_j is the characteristic vector of the movie v_j, and represents the extent to which this movie corresponds to the h factors.

To predict ratings, classic SVD technique starts by calculating the decomposition of the ratings matrix denoted M.

$$M = U\Sigma V^T \tag{10}$$

The approximation of the matrix M is given by:

$$M \approx U_h \Sigma_h V_h^T \tag{11}$$

With U_h and V_h are the reduction of the matrices U and V by keeping only the first h columns (respectively lines). By reducing the number of singular values in Σ to h, one obtains Σ_h. The vectors p_i and q_j can be calculated using the matrices Σ_h, U_h and V_h as follows:

$$p_i = U_h \Sigma_h \tag{12}$$

$$q_i = V_h \Sigma_h \tag{13}$$

Thus, the prediction of an evaluation given by a user i for a movie j will be deduced by the following equation:

$$\hat{r}_{ij} = p_i^T q_j \tag{14}$$

RegSVD calculates the vectors p_i and q_j, iteratively, where the start values are calculated using equations (12) and (13). Then, the iterative regularization algorithm [23] is implemented as follows :

$$e_{ij} = r_{ij} - \hat{r}_{ij} \tag{15}$$

$$p_{il} = p_{il} + ls * (e_{ij} q_{jl} - \lambda p_{il}) \tag{16}$$

$$q_{jl} = q_{jl} + ls * (e_{ij} p_{il} - \lambda q_{jl}) \tag{17}$$

With e_{ij} is the error associated with the prediction, p_{il} is the l^{th} value in the user's taste vector and q_{jl} is the l^{th} value in the movie's characteristic vector. ls and λ are two constants, where ls is the learning step and λ is the regularization parameter. To minimize the prediction error, the optimal experimental settings is $ls = 0.01$ and $\lambda = 0.1$ [28].

5.3.2 BiaisedMF

As a second baseline method, we use the *Biased Matrix Factorization (BiaisedMF)* technique described by Koren in [16]. This method is part of the family of collaborative filtering techniques which model $user - movie$ interactions. The principle idea behind *BiaisedMF* is that most evaluation values observed in collaborative filtering models are due to effects associated either with users or with items, regardless to their interactions. Indeed, evaluations present large distortions in user and item information, namely: systematic trends for some users to give higher ratings than other users, and for some items to receive more ratings than other items.

Koren [16] encapsulates the effects which do not reflect the $user - item$ interactions in *BiaisedMF* predictors. These predictors tend to capture the observed signal, isolate the part that truly represents $user - item$ interaction and submit it to more appropriate user preference models.

Let b_u and b_i the observed rating deviations of the user u and the item i, respectively, with respect to the average μ (the average of the observed ratings). The prediction given by *BiaisedMF* for an unknown evaluation r_{ui} is denoted by b_{ui} and represents the user and the item effects:

$$b_{ui} = \mu + b_u + b_i \tag{18}$$

5.3.3 SVD++

SVD++ [17] is an improvement of the *SVD* algorithm for recommendation, and based on the matrix factorization technique. Basic *SVD* projects items and users together in a common latent factors space of dimension f. Thus, $user - item$ interactions (evaluations) are modeled by scalar products in this space. The aim of this projection is to express evaluations by representing the items and the users using factors automatically deduced from the users' feedback [17]. Consequently, each item i is associated with a vector $q_i \in \mathbb{R}^f$, and each user u is associated with a vector $p_u \in \mathbb{R}^f$. For a given item i, the elements of q_i measure how close this item is to these factors. Elements of p_u represent the interest that the user u has in these factors. In the context of movie recommendation systems, the elements of q_i measure the degree to which a movie i belongs to cinematographic genres (corresponding to the found factors), and the elements of p_u express the interest that u has in the corresponding cinematographic genres. The resulting scalar product, $q_i^T p_u$, presents the relation between the movie i and the user u, in other words the interest that the user u has in the movie i. The *SVD* prediction rule is given by adding the product $q_i^T p_u$ to the sum of the three parameters μ, b_i and b_u:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u \tag{19}$$

With $\hat{r}_{ui}$ is the estimation of the user's u interest in the item i.

To improve the basic *SVD* algorithm for the recommendation, Koren [17] introduces an additional set of factors to the item in order to integrate the implicit feedback. Thus, Koren links each item i to a vector of factors $y_i \in \mathbb{R}^f$ iteratively calculated as follows:

$$\forall j \in |R(u)| : y_j \leftarrow y_j + \gamma(e_{ui}|R(u)|^{-\frac{1}{2}} q_i - \lambda y_j) \tag{20}$$

With $e_{ui} = r_{ui} - \hat{r}_{ui}$ the error associated with each prediction $\hat{r}_{ui}$. Elements evaluated by the user u are contained in $R(u)$. The parameters γ and λ are fixed by Koren [17] at 0.007 and 0.015 respectively.

These new factors are introduced in order to characterize users according to the evaluated elements. The new ratings prediction rule is written as follows:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \tag{21}$$

In the equation (21), the user u is represented by $p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j$, instead of p_u. Thus, the representation of the user in equation (19) is reinforced by adding the implicit feedback perspective [17].

5.3.4 MudRecS

The *MudRecS* system is proposed by Qumsiyeh and Ng [24] to predict ratings of multimedia items (movies, books or tables). This system exploits various sources of information concerning the items previously rated by a user u, in order to predict evaluations for items that u has not rated yet. The rating prediction is performed in two main steps, namely: pre-rated items analysis and evaluation prediction.

Pre-rated items analysis *MudRecS* analyzes the preferences of a user u based on article genres, review, role players (i.e. principal actor of a film, author of a book, artist of a painting or a music) and the readability level. Genres and role players of multimedia articles can be easily extracted from representative multimedia websites, such as *IMDB*[1] for movies, *last.fm*[2] for songs, *iblist*[3] for books, and *Flickr*[4] for images. Other features of multimedia articles can be recovered from reviews that

[1] http://www.imdb.com
[2] http://www.last.fm/fr
[3] http://iblist.com
[4] https://www.flickr.com/

are available at opinion websites such as: *Epinions*[5], *ConsumerSearch*[6] and *ConsumerReports*[7].

Evaluation prediction MudRecS uses the set of items rated by the user u to predict the rate that u can give to an item I not yet evaluated. Therefore, the *MudRecS* system calculates four scores for the item I, namely *Genre score (GS)*, *Review Score (RwS)*, *Role Player Score (RPS)* and *Readability Score (ReS)*.

To predict the score for each evaluation level $R_l (1 \leq l \leq N)$, *MudRecS* combines the *GS*, *RwS*, *RPS* and *ReS* scores of the item I, using the *Stanford Certainty Factor (SCF)* [19]:

$$MAX_{i=1}^{N}\{R_S(R_i, I)\} \tag{22}$$

The rating score $R_S(R_i, I)$ is given by:

$$R_S(R_l,I) = \frac{GS(R_l,I)+RwS(R_l,I)+RPS(R_l,I)+ReS(R_l,I,L)}{1-Min\_score} \tag{23}$$

Where Min_score is the lowest score among $GS(R_l, I)$, $RwS(R_l, I)$, $RPS(R_l, I)$ and $ReS(R_l, I, L)$.

If the item I is not a book, as it is our case, then the $ReS(R_l, I, L)$ score is excluded from the equation (23). The rating level R_l corresponding to the highest rating score is selected as the proposed rate for the item I.

5.4 Experimental protocol

We use the *MovieLens-100k* to evaluate our method. As mentioned in section 5.1, this dataset consists of five datasets, each of them is divided into a training set and a test set containing 80,000 and 20,000 entries, respectively. In our work, we proceed in the same way for the five datasets.

We start by creating the input matrix from each training set, as mentioned in section 4.1.1. Thus, we create a vector for each user as follows:

$$U_x = \{age, gender, occ, zip, RD, RG_1, ..., RG_{19}\} \tag{24}$$

With *age*, *sex*, *occ* and *zip* represent respectively the user's age, gender, occupation and zip-code. RD is the average of release dates of movies that user u rated. RG_i represents the average ratings that user u assigned to movies belonging to a cinematographic kind i. Therefor, U presents a user by his interest to movie genres, in addition to his personal demographic information (age, sex, etc.).

Then, by gathering the created user vectors, we construct the matrix M for each training base, where each line corresponds to a user. Then, we normalize M (as explained in section 4.1.1) to obtain the normalized matrix N, and we compute the eigenvalues and the eigenvectors of the covariance matrix of N. The obtained eigenvectors correspond to the latent centers where the cinematographic interests of the users are concentrated. Using these centers, group of users sharing the same cinematographic tastes are formed.

Finally, the training set of each dataset is divided into sub-sets, each sub-set corresponds to one of the obtained user groups. Then, using the regression algorithm and the created training sub-sets, we apply a training phase to construct, for each user group, a prediction model.

The constructed prediction models are used to predict the user interests within the identified groups. Therefore, we just identify, for each input (user u, movie i) of the test base, the group to which the user u belongs, and use the prediction model corresponding to this group to estimate the rate for the pair (u, i).

As we adopt four different regression algorithms in this work (*M5P*, *MLP*, $\nu - SVR$ and $\varepsilon - SVR$), we use different designations to distinguish between the different variants of the $GLER$ method. We denote by $GLER_{M5P}$ (respectively $GLER_{MLP}$, $GLER_{\nu-SVR}$ and $GLER_{\varepsilon-SVR}$) the method $GLER$ using the $M5P$ algorithm (respectively MLP, $\nu - SVR$ and $\varepsilon - SVR$).

We applied $GLER$ as well as the baseline methods (*RegSVD*, *BiaisedMF* , *SVD++* and *MudRecS*) on the *MovieLens-100k* training dataset in order to predict the rates that users may attribute to movies. For each method, we compute the average of *RMSE* and *MAE* values obtained on the five test datasets.

5.5 Results and discussions

Table 2 shows the MAE and $RMSE$ performances of the $GLER$ method on the five test datasets. Each column of table 2 corresponds to a variant of the $GLER$ method. The table also presents the average performance of each $GLER$ variant for the five test datasets. These results show that the average of MAE does not exceed 0.3 and the average of $RMSE$ values is lower than 0.43, regardless to the regression method used.

According to these results, we obtain the best predictions using two variants of SVR ($GLER_{\nu-SVR}$ and $GLER_{\varepsilon-SVR}$). Furthermore, the best $RMSE$ (respectively MAE) average is obtained utilizing the $\varepsilon - SVR$ (respectively $\nu-SVR$). Therefore, we can conclude that the estimated rates using $GLER_{\varepsilon-SVR}$ are mostly close to the real rates. Also, $GLER_{\nu-SVR}$ often predicts the exact rates, but the committed errors are greater than those committed with $GLER_{\varepsilon-SVR}$. The most likely

[5] http://www.epinions.com
[6] http://www.consumersearch.com
[7] http://www.consumerreports.org

Table 2 Rating prediction accuracy of the four variants of *GLER*.

		$GLER_{M5P}$	$GLER_{MLP}$	$GLER_{\varepsilon-SVR}$	$GLER_{\nu-SVR}$
Dataset 1	MAE	0.28	0.29	0.30	0.29
	RMSE	0.42	0.40	0.38	0.39
Dataset 2	MAE	0.31	0.29	0.30	0.28
	RMSE	0.42	0.40	0.39	0.40
Dataset 3	MAE	0.32	0.28	0.30	0.28
	RMSE	0.44	0.44	0.39	0.39
Dataset 4	MAE	0.31	0.29	0.30	0.28
	RMSE	0.44	0.40	0.38	0.39
Dataset 5	MAE	0.29	0.31	0.30	0.28
	RMSE	0.41	0.44	0.38	0.39
Average	**MAE**	**0.30**	**0.29**	**0.30**	**0.28**
	RMSE	**0.43**	**0.42**	**0.38**	**0.39**

explanation of these results is that $GLER_{\varepsilon-SVR}$ aims at limiting the amount of errors in the model and to seek the best performance without controlling the support vectors in the resulting model, in contrast to the algorithm $GLER_{\nu-SVR}$.

Table 3 illustrates a comparison of the accuracy results of our method *GLER* and the four baseline methods on the *MovieLens-100k* dataset. The results show that the performance of the *GLER* method exceeds the performance of the baseline methods. Indeed, compared to the best accuracy of the baseline methods, given by *MudRecS*, *GLER* decreases the *MAE* of about 0.19 and drops the *RMSE* values from 0.72 to 0.38.

Table 3 Comparison of the results obtained by *GLER* against those obtained by the baseline methods.

		MAE	**RMSE**
RegSVD		0.73	0.93
BiaisedMF		0.71	0.91
SVD++		0.71	0.90
MudRecS		0.47	0.72
GLER	$M5P$	0.30	0.43
	MLP	0.29	0.42
	$\varepsilon-SVR$	0.30	0.38
	$\nu-SVR$	0.28	0.39

The predictions given by *RegSVD*, *BiaisedMF* and *SVD++* are mainly based on three information: the average of ratings given by the user, the average of ratings given to the movie and the overall average of ratings. These three information cause some problems and affect the prediction that explains the low results. Indeed, using the average of the user's ratings limits the number of entries in the learning phase and reduces the range of items to recommend. In addition, if the history of user's ratings contains predominantly high values, the predicted ratings will tend to be elevated regardless of the movie genre. Idem for ratings given to movies, if the history of the movie's evaluation contains mostly high values, the predicted ratings will tend to be high Independently of the user cinematographic taste.

In addition, the average score for the movie to be recommended and the overall average of the recorded ratings are calculated using ratings from all users, regardless of their cinematographic tastes. Thus, these values will be influenced by the judgment of users having different tastes from that of the user concerned by the recommendation.

The *MudRecS* method combines three information for the rating prediction task, namely: the gender score, the review score and the role player score (the readability score is excluded because the recommended items are not books). In the three scores, *MudRecS* primarily seeks the preferences of the user u for the genres, characteristics and role players in the history of ratings that u has assigned. Even in the case of the review score, the ratings of other users are used to identify the item characteristics, but the rating is calculated based on the user u ratings assigned to the items according to the identified characteristics. Therefore, the *MudRecS* system is limited to the ratings of the user u and ignores the judgment of other users which is an important information.

GLER method solves the problems that the baseline methods suffer from, by enriching the rating information with the experience (the rating history) of users having similar cinematographic tastes. In addition, the input matrix used in the *GLER* method presents the users interests in movie genres and not the user-movie interactions. Thus, our method is more flexible and is able to predict ratings for movies even if they are not part of the training dataset, which is not valid for the majority of the baseline methods. Moreover, unlike the baseline methods, we enrich the rating information with user's demographic data, which is very rich in information and refines the prediction accuracy.

By using the *PCA* technique to group like-tasted users, we overcome the sparseness problem that characterizes the data handled by the recommendation systems. Also, we reduce the data size, which is very important and very useful to construct the prediction model.

Finally, we can conclude that the creation of user groups according to their cinematographic tastes and the use of these groups to predict ratings improves the performance. Indeed, the prediction will be based, not only on the history of the user ratings, but also on ratings of other users having the same cinematographic tastes. Furthermore, using the "divide and conquer" principle, creating similar user groups facilitates the task of regression algorithms. Indeed, creating a model for a small number of users having the same interests is easier and leads to more accurate predictions.

The findings of this work have to be seen in light of some limitations. First, the data quality may affect the prediction results. Indeed, one of the characteristics of the Movielens datasets is that the users change their rating behavior over time, which affect their actual preferences. Second, the used dataset includes only successful users (users having at least 20 recorded ratings), which can be considered as bias. In fact, there are many users with different rating behaviours and different preferences, which are not included in the dataset. Third, the proposed method does not present a solution for the cold start problem, which occurs when a new user with no recorded ratings has been registered. The proposed method may present a partial solution for the cold start problem by assigning the new user to the nearest user group, according to his demographic information. However, this solution needs to be proved.

6 Conclusion

In this paper, we presented a method to estimate ratings in recommendation systems, especially in the movie recommendation task. *GLER* begins by creating user groups that share the same interests (cinematographic tastes) based on their recorded ratings for movies. Then, it uses these groups to construct prediction models specific to each user group. Each constructed model will then be used to estimate ratings of users within the corresponding group.

To evaluate the proposed method, we used the MovieLens - 100k dataset, and compared its performances against four baseline algorithms, namely: *RegSVD*, *BiaisedMF*, *SVD++* and *MudRecS*. The obtained results show that the predictions made by our method are closer to the right ratings and that we obtain an improvement in *MAE* and *RMSE* of about 0.19 and 0.34 respectively. In our method, we used three regression algorithms (*M5P*, *MLP* and *SVR*) to construct the recommendation model and infer ratings. Experiments show that the two alternatives of SVR ($\nu - SVR$ and $\varepsilon - SVR$) have the greatest accuracy.

In future work, we plan to seek a new solution for the cold start problem to assign new users to groups. One of possible solutions is to exploit information from other social networks to find users with similar interest. Indeed, a new category of social networks such as *Circleme*[8], *LikeMind*[9] and *Affimity*[10] have became popular. These social networks allow users to discover people with similar interests, opinions and ideas. Also, the idea of exploiting the time dimension to regroup users is very interesting, as their interest may change over time.

References

1. Aivazoglou M, Roussos AO, Margaris D, Vassilakis C, Ioannidis S, Polakis J, Spiliotopoulos D (2020) A fine-grained social network recommender system. Social Netw Analys Mining 10(1):8, DOI 10.1007/s13278-019-0621-7
2. Alam S, Dobbie G, Riddle P, Koh YS (2012) Hierarchical pso clustering based recommender system. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2012, Brisbane, Australia, June 10-15, 2012, pp 1–8
3. Amatriain X, Jaimes A, Oliver N, Pujol JM (2015) Data mining methods for recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender Systems Handbook, Springer US, pp 227–262
4. Brugger C, Chinazzo AL, John AF, de Schryver C, Wehn N, Schlauch WE, Zweig KA (2016) Increasing sampling efficiency for the fixed degree sequence model with phase transitions. Social Netw Analys Mining 6(1):100:1–100:14, DOI 10.1007/s13278-016-0407-0
5. C Aggarwal C (2016) Neighborhood-Based Collaborative Filtering, pp 29–70
6. Das J, Mukherjee P, Majumder S, Gupta P (2014) Clustering-based recommender system using principles of voting theory. 2014 International Conference on Contemporary Computing and Informatics (IC3I) pp 230–235
7. Feng M, Loy LY, Zhang F, Zhang Z, Vellaisamy K, Chin PL, Guan C, Shen L, King NKK, Lee KK, Ang BT (2012) Intracranial Pressure and Brain Monitoring XIV, Springer Vienna, chap Go Green! Reusing Brain Monitoring Data Containing Missing Values: A Feasibility Study with Traumatic Brain Injury Patients, pp 51–59
8. Freyne J, Berkovsky S, Smith G (2011) User Modeling, Adaption and Personalization: 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, chap Recipe Recommendation: Accuracy and Reasoning, pp 99–110

[8] http://www.circleme.com
[9] http ://likemind.us
[10] https ://affimity.com

9. Goldberg K, Roeder T, Gupta D, Perkins C (2001) Eigentaste: A constant time collaborative filtering algorithm. information retrieval 4(2):133–151
10. Guan X, Li CT, Guan Y (2016) Enhanced svd for collaborative filtering. In: Proceedings, Part II, of the 20th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume 9652, Springer-Verlag, Berlin, Heidelberg, PAKDD 2016, pp 503–514
11. Huang S, Li X, Candan KS, Sapino ML (2016) Reducing seed noise in personalized pagerank. Social Netw Analys Mining 6(1):6:1–6:25, DOI 10.1007/s13278-015-0309-6
12. Idrissi N, Zellou A (2020) A systematic literature review of sparsity issues in recommender systems. Social Netw Analys Mining 10(1):15
13. Jaffali S, Jamoussi S (2012) Principal component analysis neural network for textual document categorization and dimension reduction. In: 2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), pp 835–839
14. Jaffali S, Jamoussi S, BenHamadou A, Smaili K (2016) Grouping like-minded users for ratings' prediction. In: Czarnowski I, Caballero AM, Howlett RJ, Jain LC (eds) Intelligent Decision Technologies 2016, Springer International Publishing, Cham, pp 3–14
15. Jelassi MN, Ben Yahia S, Mephu Nguifo E (2015) Towards more targeted recommendations in folksonomies. Social Network Analysis and Mining 5(1):68:1–68:18
16. Koren Y (2008) Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, USA, KDD '08, pp 426–434
17. Koren Y, Bell RM (2015) Advances in collaborative filtering. In: Recommender Systems Handbook, pp 77–118
18. Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7(1):76–80
19. Luger GF (2008) Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 6th edn. Addison-Wesley Publishing Company, USA
20. Margaris D, Vassilakis C, Spiliotopoulos D (2019) Handling uncertainty in social media textual information for improving venue recommendation formulation quality in social networks. Social Netw Analys Mining 9(1):64:1–64:19, DOI 10.1007/s13278-019-0610-x
21. Mnih A, Salakhutdinov R (2007) Probabilistic matrix factorization. NIPS pp 1257–1264
22. Ning X, Desrosiers C, Karypis G (2015) A comprehensive survey of neighborhood-based recommendation methods. In: Recommender Systems Handbook, pp 37–76
23. Paterek A (2007) Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, vol 2007, pp 5–8
24. Qumsiyeh R, Ng YK (2012) Predicting the ratings of multimedia items for making personalized recommendations. In: The 35th International ACM SIGIR conference on research and development in Information Retrieval, Portland, USA, SIGIR 12, pp 475–484
25. Raiche G, Walls TA, Magis D, Riopel M, Blais J (2013) Non graphical solutions for the cattell's scree test. In: Journal of Research Methods for the Behavioral and Social Sciences, vol 9, pp 23–29
26. Sarwar BM, Karypis G, Konstan JA, Riedl JT (2000) Application of dimensionality reduction in recommender system – a case study. In: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Web Mining for E–Commerce, USA, pp 1–12
27. Scholkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New support vector algorithms. Neural Computation 12(5):1207–1245
28. Taheri SM, Mahyar H, Firouzi M, K EG, Grosu R, Movaghar A (2017) Extracting implicit social relation for social recommendation techniques in user rating prediction. In: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017, pp 1343–1351
29. Zahra S, Ghazanfar MA, Khalid A, Azam MA, Naeem U, Prugel-Bennett A (2015) Novel centroid selection approaches for kmeans-clustering based recommender systems. Inf Sci 320:156–189
30. Zhang W, Wang J (2015) A collective bayesian poisson factorization model for cold-start local event recommendation. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '15, pp 1455–1464