Managing Natural Noise in Collaborative Recommender Systems

Raciel Yera Toledo
Knowledge Management Center
University of Ciego de Ávila
Carretera a Morón Km 9 ½,C.A, Cuba
ryera@cegic.unica.cu

Luis Martínez López
Department of Computer Science
University of Jaen
23071- Jaen, Spain
luis.martinez@ujaen.es

Yailé Caballero Mota
Department of Computer Science
University of Camagüey
Circun. Km 5 ½ , Camagüey, Cuba
yaile.caballero@reduc.edu.cu

***Abstract* – Recommender systems help users to find information that best fits their preferences and needs in an overloaded search space. Most of recommender systems research focuses on improving recommendation methods to obtain a higher accuracy in recommendations. However, the study of user's inconsistencies, so-called *natural noise,* is becoming a hot topic in Recommender Systems. In this contribution is proposed a novel approach to detect and correct those inconsistent ratings that might bias recommendations, by using global information about user and item preferences. This proposal characterizes items and users by their ratings and classifies a rating as noisy if it contradicts user or item tendencies. This approach just utilizes ratings on the contrary of previous proposals that use additional information like item attributes or user interaction.**

I. Introduction

Recommender systems have become a mainstream research field in information technologies. They appear as a solution to help users to obtain information that best fits their preferences and needs, in scenarios in which information overloading is an important drawback.

The most popular version of this kind of applications [1], *learns* from user preferences about a predefined set of known items, and predicts the preference degree for this user about an unknown item. With this goal, an important amount of applications have been built to recommend different items like movies, books, TV shows, jokes, news, scientific papers, web pages, and so on. Some application areas have been e-commerce and e-learning [2, 3].

Most of recommender systems are developed by using two main paradigms: content-based and collaborative filtering. Content-based recommender systems [4, 5], suggest items with similar features, considering those ones that the user preferred in the past. On the other hand, collaborative filtering recommender systems [6, 7] recommend items that other users with similar tastes liked in the past. Collaborative filtering systems can generate recommendations by using just information about user preferences over a group of items. For this reason, they become popular nowadays.

Lots of work have been done in collaborative filtering to improve recommendation accuracy [6, 7]. It includes user-user and item-item approaches, dimensionality reduction approaches, probabilistic methods, MDP- based methods, graph-based methods, rule-based methods, and hybrids [8]. In past years, advances in collaborative filtering [9] have overcome important challenges like Netflix Prize, showing that rating prediction can be significantly improved.

In practical data mining problems usually the data contains a degree of inconsistency, so preprocessing is an important step in mining processes [10]. However, most of works done in recommendation problems assumes that rating data are free of irregularities. Nevertheless, recently Amatriain, Pujol and Oliver [11] showed that users could be inconsistent when they rate items, making recommender systems data vulnerable to inconsistencies. These inconsistencies, so called *natural noise* [12], are inserted without malicious intentions, and its manipulation represents an open problem to improve recommender systems performance [6, 13, 14].

In this contribution, we will propose a way to explore datasets and apply collaborative filtering techniques to correct noisy ratings. This proposal considers that user and item are represented by their rating profiles, and classifies them according to it. After that, a rating is considered as noisy if it is contradictory with the user/item classification. In such a case, it is then corrected by using a strategy. To validate the hypothesis that corrected data improve the accuracy of collaborative filtering recommender systems, a case study on Movielens dataset has been carried out.

The paper is organized as follows: Section 2 reviews collaborative filtering and relevant work on natural noise and rating correction. Section 3 introduces our approach for managing natural noise in two steps, first noisy rating detection and then noise correction. Section 4 performs and discusses a case study to evaluate the proposal and finally section 5 concludes the contribution.

II. BACKGROUND

In this section a brief review about collaborative filtering and natural noise is introduced.

A. *Collaborative Filtering*

Collaborative filtering recommender systems (CFRS) provide recommendations to users about items that have previously liked to people with similar tastes. While others recommender systems approaches depend on content directly associated to items, CF approach generally uses just ratings to predict user's preferences over items. The data in CF could be gathered in an explicit or implicit way [3].

To evaluate our proposal, in our case study we will use three classical approaches for collaborative filtering [8]:

a) *Memory based user-user collaborative filtering* [15]. It predicts user's preferences for a not rated item, looking for other similar users to the active user. Neighbors' ratings for the corresponding item are weighted considering similarity degrees, to predict unknown user's preference. Similarity between two users is calculated using different similarity measures.

b) *Model based item-item collaborative filtering* [16]. It does not use similarities between users to predict preferences. It uses similarities between items that stores in a model. To predict an unknown preference for a current user and an item, it takes from the model similar items and uses user's rating from those items to predict an unknown rating.

c) *Matrix factorization approach.* It appears as a solution for classical scalability and sparsity problems in traditional memory-based collaborative filtering. It proposes a user and item space reduction to a lower dimensionality space that retains most of the initial information, mitigating redundancy and sparsity effects. This new k-space includes a set of k topics in which user preferences can be expressed considering user's interest in a topic, and the degree to which each item is significant to the topic. Early works like [17] formally use latent semantic analysis in the form of truncated singular value decomposition (SVD) to reduce dimensionality in CF.

However, in the last years less expensive and more accurate methods have been developed using alternative procedures to reduce dimensionality. Those proposals, summarized in [9], focus on directly learn a new space using gradient descendent techniques, instead of transform the initial rating matrix in a new one. Our case study, presented in section IV, uses the framework introduced in [9].

B. *Handling Natural Noise in Collaborative Filtering*

Noise in recommender systems dataset has been grouped in two main categories:

1) *Malicious noise*, associated to noise intentionally introduced by an external agent to bias recommender results [18], and

2) *Natural noise*, involuntarily introduced by users in recommender dataset that could also affect the recommendation result.

While handling malicious noise is a well-developed research area, natural noise has received less attention. Note that both noises are produced by different sources, so techniques to process them must be different.

Recently, Amatriain et al. [11] consider that natural noise characterization is a key element in recommender systems. They made some experiments to quantify users inconsistencies degree analyzing their ratings, and concluded that this noise can considerably affect recommendation quality.

O' Mahony et al. [12] present one early work that uses natural and malicious noise terms. It detects noisy ratings comparing each rating with user's predicted preference for that item. These predictions are calculated using a set of *genuine* user profiles manually obtained. It then discards ratings which differences exceed a threshold considering this comparison process. Ekstrand et al. in [6], point out that this approach makes the erroneous assumption that all high-deviance ratings are result of user mistakes, and it does not consider the presence of some outliers that express the current user preference.

An interactive method to eliminate noisy ratings, named item re-rating, was proposed in [13]. When any noisy rating is discovered, the system decides to ask the user to rate the corresponding item again, and the author considers that this re-rating process will remove the natural noise appeared. It affirms that denoising extreme ratings yields larger performance gain than denoising mild ratings. Noisy ratings and noisy users are previously identified in online rating trials done by authors as a part of its work. The mandatory user participation is an important component in this method, and transforms it in impractical in some scenarios.

Recently, another important group of works explores data beyond ratings to correct erroneous rating values in recommender systems [19]. They use item attributes to learn a user preference model, and it marks a rating as incorrect if it belongs to the current user preference model, but it is under a predefined threshold and under mean rating for this user. They also propose a way to correct rating, defining some criteria to classify a user as expert, and proposing three rating-correction methods using experts. The first one considers a weighted average rating of experts about the same item. The second one considers a weighted average rating of items belonging to the user preference model (similarly to item-to-item collaborative filtering), and the third one calculates mean value of previous two. This approach improves recommender performance, but depends on item attributes to build the preference model. In the current contribution, we also correct erroneous values, but without using external sources.

Eventually, Li [20] proposes an approach to process natural noise in recommender systems. It detects "noisy but non-malicious" users in CFRS, assuming that the ratings provided by a user on closely correlated items should have similar scores. For each user profile, it quantifies underlying noise, obtaining users with a high noise degree. Removing these users for dataset, recommendation accuracy is improved. However, this work focus on noise detection on user level, so new approaches beyond users, that deals with rating levels, are needed. Noise treatment at this deeper level will facilitate rating correction, avoiding information removal from the dataset like is demanded in [20].

III. MANAGING NATURAL NOISE BY USING RATINGS

Here, it is presented a novel approach to detect and correct noisy ratings, taking into account just rating values. It does not use any additional information about users or items.

Disregarding preferences variation over time, it is considered that erroneous ratings can appear in recommender systems dataset due to several reasons: i) Users unintentionally can express information that does not correspond with their preferences and profiles. This is the classical scenario for natural noise. ii) They also could insert anomalous-but-correct information intentionally added, that is not actually aligned with their global profiles and could be considered as noisy.

For example, if a user likes all Woody Allen's films except just one, this anomalous preference could associate user with a different group of users that they actually do not belong, and the recommendations that could receive might be influence from this group. Disregarding the doubt about this rating and if it represents real preferences or not, it certainly affects user predictions regarding unknown items. Our research focuses on processing this kind of *incorrect* ratings, but we lack of additional data like "Woody Allen", so we need to define new strategies to do it just using ratings.

In all cases, erroneous ratings or ratings that do not represent real user's preferences can cause accuracy decay in collaborative filtering. *Incorrect* ratings could alter user's profile, and the *biased* user could fall in a different neighborhood comparing with *unbiased* one. This could affect current user's predictions, and also predictions for users in the neighborhood.

In order to correct natural noisy ratings, we propose a framework that includes two steps:

1) *Noisy ratings detection*: it classifies user and item profiles considering their ratings into four different classes. Each rating can also be classified into three different classes. A rating is marked as possible noise, if exist contradictions between the rating, considering its class, and the classes associated to the current user and item.

2) *Noise correction*: it uses a basic collaborative filtering method to predict a new rating for each possible noisy one. If the difference between both ratings is greater than a predefined threshold, the old rating is definitively considered as noisy, and its value is replaced with the new one.

A further detailed description of previous phases is introduced below:

A. Noisy Rating Detection

To detect natural noisy ratings, we initially perform a classification for ratings, users, and items. We assume that each user has his/her own tendency when rates items: i) A group of users tends to positively evaluate all items, ii) another group provides average values, iii) a third one usually gives low ratings, and also, iv) there is a fourth group which behavior oscillates among any of the former categories, and do not fall into a specific one.

This classification is also extended to items. i) There is a group that is highly preferred by all users, ii) a group which items is averagely preferred, iii) a group that is not preferred by the majority of users, and like in users, and iv) a group that contains items with divided opinions about their preferences degree.

Eventually, each rating r(u,i) (for a user *u* and an item *i*) is classified into three different classes according its value:

Rating classes:

1. *Weak preference*: A rating r(u,i) is classified as a weak preference if it verifies $r(u,i) < \kappa$.
2. *Average preference*: A rating r(u,i) is classified as an average preference if it verifies $\kappa \leq r(u,i) < \nu$.
3. *Strong preference*: A rating r(u,i) is classified as a strong preference if it verifies $r(u,i) \geq \nu$.

This classification depends on a weak-average threshold κ and an average-strong threshold ν that must be previously defined, satisfying $\kappa < \nu$. We suggest later on a procedure to calculate tentative values for these parameters.

Preferences for each user *u* can be grouped in sets W_u, A_u and S_u, and for each item *i* in sets W_i, A_i and S_i, considering U and I as whole sets of users and items:

User and item sets:

1. W_u={r(u,i) | } $\forall$ i $\in$ I where r(u,i) is a weak preference}
2. A_u={r(u,i) | } $\forall$ i $\in$ I where r(u,i) is an average preference}
3. S_u={r(u,i) | } $\forall$ i $\in$ I where r(u,i) is a strong preference}

1. W_i={r(u,i) | } $\forall$ u $\in$ U where r(u,i) is a weak preference}
2. A_i={r(u,i) | } $\forall$ u $\in$ U where r(u,i) is an average preference}
3. S_i={r(u,i) | } $\forall$ u $\in$ U where r(u,i) is a strong preference}

W_u is the set of ratings provided for user *u* that represents weak preferences, A_u as the set of average preferences for user *u*, and S_u as the set of strong preferences for user *u*. Item sets are similar.

Considering rating classes and user and item sets, user profiles can be formally classified into four different categories: *benevolent, average, critical*, and *variable*. On the other hand, items can be classified in four categories: *strongly-preferred, averagely-preferred, weakly-preferred* and *variably-preferred* (see Table I).

They are classified in one of the first three categories in each case, if the amount of ratings belonging to the corresponding class rating (following rating classes) exceeds the amount of the others two. Specifically, for each user and item, it is counted the amount of ratings belonging to each preference class and then:

i) If ratings representing weak preferences exceed both ratings representing average and strong preferences, then user is classified as critical, or item is classified as weakly-preferred.

ii) If average preferences exceed the sums of other two, then user is classified as average, or item as averagely-preferred.

iii) If strong preferences verify the same condition, then user is classified as benevolent, or item as strongly-preferred.

iv) If there is no amount that exceeds the sum of other two, user or item is classified as variable.

TABLE I
USER AND ITEM CLASSES PROPOSED

User classes	
Critical user	Verifies card(W_u) ≥ card(A_u) + card(S_u)
Average user	Verifies card(A_u) ≥ card(W_u) + card(S_u)
Benevolent user	Verifies card(S_u) ≥ card(W_u) + card(A_u)
Variable user	Does not satisfy the others user conditions
Item classes	
Weakly-preferred item	Verifies card(W_i) ≥ card(A_i) + card(S_i)
Averagely-preferred item	Verifies card(A_i) ≥ card(W_i) + card(S_i)
Strongly-preferred item	Verifies card(S_i) ≥ card(W_i) + card(A_i)
Variably-preferred item	Does not satisfy the others item conditions

Table I summarizes user and item classification. It considers function *card(A)* that represents the cardinality of the corresponding set.

Once each rating, user and item has been classified, the detection process looks for contradictions among them. In general, it is assumed that if a rating belongs to similar classes regarding its user and item, then it must belong to the similar rating class. If not, the rating could be erroneous.

Specifically, to determine if a rating could be noisy is presented the following procedure:

1. Classify the rating, and classify its corresponding user and item.

2. Mark it as possible noise if:

a) User class is critical, item class is weakly-preferred, and rating class is an average or a strong preference.

b) User class is average, item class is averagely-preferred, and rating class is a weak or a strong preference.

c) User class is benevolent, item class is strongly-preferred, and rating class is an average or a weak preference.

B. Noise Correction Process

Some authors propose to discard noisy ratings when it predicts new ratings [12], while others consider that these ratings must be corrected [19]. This contribution adopts the latter view. Before correction, we verify another reason that classifies ratings as noise. For each marked value as possible noise, it is predicted another value using traditional memory-based user-user collaborative filtering with Pearson's similarity and k=60 [1], and we compare it with the current value. If the difference between them exceeds a predefined threshold δ then the new value is set as rating. Otherwise, the initial one is kept.

Summarizing, the whole correction process is:

```
1. Classify each user and item according definitions.
2. For each rating
    2.1. Classify them according definition.
    2.2. Following the procedure presented in the previous
         section, mark it if represents a possible noise.
3. For each rating marked as possible noise
    3.1. Predict its value using traditional collaborative
         filtering
    3.2. Calculate the difference between predicted and
         original value.
        3.2.1. If difference exceeds a threshold, then
               substitute the original with the predicted value.
        3.2.2. Otherwise, remain the value as the original one.
```

C. On Parameter Values

Our proposal depends on three parameters: a weak-average threshold (κ), an average-strong threshold (ν), and a prediction difference threshold (δ). These parameters are highly domain-dependant, so it is difficult to predetermine its optimal values. However, it could be defined a strategy to assign them good initial values.

Considering that ratings are ordinal values and that we assume three possible classes for ratings, we suggest to select values for classification thresholds (κ and ν) in a way that approximately divide the rating range into three equal bins.

Equations (1) and (2) allow to calculate values for κ and ν following these criteria, considering the use of *ent(n)* as a function to determine the immediately-inferior entire value of *n*, to assure that thresholds receive entire values; and using minimum and maximum possible values for rating.

$$\kappa = \min Rating + ent(\frac{1}{3} * (\max Rating - \min Rating)) \quad (1)$$

$$\nu = \max Rating - ent(\frac{1}{3} * (\max Rating - \min Rating)) \quad (2)$$

On the other hand, considering that rating values are represented in an ordinal scale, we suggest to assign the value of a minimum step in this scale, to the difference threshold (δ). We assume that if the difference between new and old rating values exceeds the minimum difference between two consecutive values in the rating scale, then it is enough larger and then the replacement must be performed.

We leave to future works the proposal of alternatives strategies to obtain the initial parameter values in our method.

IV. Case Study

In the previous sections, we have presented an approach to correct natural noise in CFRS. In order to obtain the effects of our proposal, we carry out a case study over Movielens, which is a popular dataset in recommender systems field. Movielens original version is a well-known dataset containing 100000 movie ratings on 943 users and 1682 items where each rating is discrete and is in the range [1,5].

We used the experimental protocol suggested by Gunawardana and Shani in [21]. It proposes to select a set of users from the original dataset. Then, randomly select the amount of items n_u to hide for each user u, and finally hide n_u items at the corresponding u. These hidden items will conform test set, and remaining will be the training set. We repeat this process several times, conforming several training set-test set pairs to be used, in order to measure experiment consistency. In this work, for each dataset, we will consider five pairs.

The referred work proposes for each test rating, to make a new rating prediction for the corresponding user and item, using algorithm to evaluate and training data. Algorithm´s quality is calculated using mean absolute error (MAE) (3) over all predictions made.

$$MAE(f) = \frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} |f(u,i) - r_{ui}| \quad (3)$$

Gunawardana and Shani also show a way to statistically compare two recommendation algorithms. It suggest to use a user approach, where for each user in test set we calculate two MAE values representing average of their predictions, made using each algorithm. For *n* users, we will obtain two sets with predictions, and then a Wilcoxon signed test between these distributions will inform us about statistical difference between algorithms. In order to measure our correction process effects, we run the correction process over training set, and after it, we compare traditional algorithm performance predicting test set ratings, using transformed data and using original one.

Both training and test set can contain noise. So, considering all ratings in test set to calculate MAE could affect final experiment results. For this reason, before evaluation, we exclude from test set, those ratings that also could be possible noise. With this purpose, for each test rating we classify it according definition, and verify if it has contradictions considering corresponding user and item classification determined with training data (sections III.A and III.B). If contradictions exist, then rating is considered as noisy, and we exclude it from test set.

For the mentioned dataset, we firstly show information about correction process, like the amount of users and items per class, the amount of possibly noisy ratings, and the amount of finally corrected ratings. As a main part, we measure the behavior, in MAE terms, of three methods explained in section II (user-user, item-item, and a biased SVD matrix factorization approach), with and without rating correction. We also refer statistical processing results. In our experiments, we use biased SVD implementation provided by MyMediaLite. It allows to specify two different regularization extent values (λ) for vectors and for biases. Considering other aspects, it exactly implements model mentioned in section 2.

Considering that our experiments goal is to determine correction effect in recommendation accuracy, we fix some parameters in biased SVD matrix factorization approach for all experiments. Those parameters were regularization (λ_1=0.015), bias regularization (λ_2=0.015) and learning rate (α=0.01). Initial user and item values were initialized considering init_mean=0 and init_stdev=0.1. We appreciate that different values on these parameters do not directly affect our comparison proposal.

To prepare data for experiments, we randomly selected 900 users, and conforms training and test set as we explained before. We perform this task five times, selecting each time a different set with the same amount of users.

Following suggestions in section III, we set values κ=2 for weak-average threshold and ν=4 for average-strong threshold. Considering that ratings in Movielens are in the range [1, 5] with step 1, we assign δ=1 for difference threshold between predicted and original rating.

TABLE II
AVERAGE AMOUNT OF USERS AND ITEMS PER CLASS, RATINGS WITH POSSIBLE NOISE, AND RATINGS FINALLY CORRECTED IN MOVIELENS

Users per class			
Critical users	Average users	Benevolent users	Variable users
6.4	178.6	572.2	142.8
Items per class			
Weakly-preferred item	Averagely-preferred item	Strongly-preferred item	Variably-preferred item
89.4	541.8	593.8	333.2
Amount of possible noisy ratings after classification		Amount of finally corrected ratings	
5362 (11,27 % of ratings in training set)		2000 (4,20 %)	

Table II presents the average amount of users and items belonging to each defined class, for all partition obtained. It shows that the amount of users and items per class, slightly vary depending on the corresponding case, but it has an important consistency degree. We can observe that an important user and items quantity belongs to the benevolent and strongly-preferred classes respectively. Also, there are few users with a critical behavior. On the other hand, Table II shows the average amount of possible noisy ratings in training set after classification, and the amount of finally corrected noisy ratings.

TABLE III
MAE VALUES WITH AND WITHOUT RATING CORRECTION (PEARSON'S USER-USER AND ITEM-ITEM COLLABORATIVE FILTERING)

	User-User CF	User-User CF + rating correction	Item-Item CF	Item-Item CF + rating correction
k=10	0.7143	**0.6961**	0.7179	**0.6959**
k=20	0.7054	**0.6887**	0.7111	**0.6909**
k=30	0.7039	**0.6874**	0.7100	**0.6901**
k=40	0.7037	**0.6873**	0.7099	**0.6900**
k=50	0.7037	**0.6873**	0.7098	**0.6900**
k=60	0.7039	**0.6875**	0.7098	**0.6901**

In order to compare user-user Pearson´s collaborative filtering behavior against user-user behavior with data correction, for each partition we perform several experiments modifying the amount of nearest neighbor in collaborative filtering (k value). Rating correction always implies a MAE improvement, comparing with the approach that does not consider correction, and this result is statistically significative considering the referred procedure using Wilcoxon test **($p<0.05$).** We also run the same protocol for item-item Pearson's collaborative filtering, obtaining the same improvement. Both results are showed in Table III.

TABLE IV
MAE VALUES WITH AND WITHOUT RATING CORRECTION (MATRIX FACTORIZATION APPROACH)

Matrix factorization approach	Matrix factorization approach + rating correction
0.7008	**0.6837**

In the case of matrix factorization approach, at first we empirically obtained values for the amount of factors in user and item vectors, and the amount of iterations needed. We globally obtain best results for num_factors=5 and num_iters=20. Using those values, we run algorithm for each partition, and averaged the results. This process was run several times, considering that this approach is not a deterministic one. Table IV shows general average values, concluding that correction process significantly improve accuracy recommendation considering this popular dimensionality reduction approach. Statistical processing corroborate this sentences, with **$p<0,05$.**

Summarizing, the proposed approach obtained positive results for the datasets used in experiments. It shows that benevolent users represent the majority of users, and strongly-preferred items represent the majority of items in the case study developed. In general, results show that our classification approach fits well in recommender system's dataset, because it finds users and items for all classes. It also proves that our correction approach decrease MAE value disregarding the algorithm used. We remark that still for a dimensionality reduction method like biased matrix factorization approach, which focus on remove small disturbances, decreasing the impact of noise [6]; our correction process improves MAE value in a similar degree comparing with the other two traditional recommendation methods (Pearson's user-user and item-item).

V. CONCLUSIONS

This contribution provides a novel approach to deal with *natural noise* in Recommender Systems in order to improve recommendations. This approach does not need any additional information to improve recommendations, because it just uses the ratings provided by users. This is an important improvement regarding previous approaches in this topic.

Our approach manages natural noise by using a *detection process* that classifies items, users and ratings looking for noisy ratings and a *correction process* that replaces the noisy ratings if necessary. The results obtained by this approach show the importance of managing natural noise to improve Recommender Systems performance.

ACKNOWLEGMENT

This work is partially supported by the Research Project TIN-2012-31263 and FEDER funds

REFERENCES

[1] G. Adomavicius, Tuzhilin, A., "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, pp. 734-749, 2005.

[2] K. J. B. Schafer, J.A., Riedl, J., "E-commerce recommendation applications," *Data Mining and Knowledge Discovery,* vol. 5, pp. 115-153, 2001.

[3] E. J. Castellano, Martínez, L. , "A Web-Decision Support System based on Collaborative Filtering for Academic Orientation. Case Study of the Spanish Secondary School.," *Journal of Universal Computer Science,* vol. 15, pp. 2786-2807, 2009.

[4] P. Lops, De Gemmis, M., Semeraro, G, "Content-based recommender systems: state of theart and trends.," in *Recommender systems handbook*, L. R. F. Ricci, B. Shapira, P.B. Kantor, Ed., ed: Springer, 2011, pp. 73-105.

[5] L. Martínez, Pérez, L.G., Barranco, M.J., "A multigranular linguistic content-based recommendation model.," *International Journal of Intelligent Systems,* vol. 22, pp. 419-434, 2007.

[6] M. D. Ekstrand, Riedl, J. T., Konstan, J. A., "Collaborative filtering recommender systems. ," *Foundations and trends in Human-Computer Interaction,* vol. 4, pp. 81-173, 2010.

[7] F. Ricci, Rokach, L., Shapira, B, Kantor, P.B., *Recommender Systems Handbook*: Springer Science+Business Media, 2011.

[8] X. Su, Khoshgoftaar, T., "A survey of collaborative filtering techniques," *Advances in artificial intelligence,* vol. 2009, p. 19, 2009.

[9] Y. Koren, Bell, R. M. Volinsky, C., "Matrix factorization techniques for recommender systems.," *IEEE Computer,* vol. 42, pp. 30-37, 2009.

[10] J. Han, Kamber, M., *Data Mining: concepts and techniques. (2nd ed.).* San Francisco, 2006.

[11] X. Amatriain, Pujol, J., Oliver, N. , "I like it... I like it not: Evaluating user ratings noise in recommender systems," presented at the 17th International Conference on User Modeling, Adaptation and Personalization (UMAP), 2009.

[12] M. P. O'Mahony, Hurley, N.J., Silvestre, G.C. , "Detecting noise in recommender system databases.," presented at the 11th ACM International Conference on Intelligent Users Interfaces (IUI) 2006.

[13] X. Amatriain, Pujol, J., Tintarev, N., Oliver, N., "Rate it again: Increasing recommendation accuracy by user re-rating.," presented at the 3rd ACM International Conference on Recommender Systems (RecSys), 2009.

[14] A. Said, Jain, B.J., Narr, S., Plumbaum, T., "Users and Noise: The Magic Barrier of Recommender Systems," presented at the 23th International Conference on User Modeling, Adaptation and Personalization (UMAP '12), 2012.

[15] P. Resnick, Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J., "Grouplens: an open architecture for collaborative filtering of netnews," presented at the ACM Conference on Computer Supported Cooperative Work, 1994.

[16] G. Linden, Smith, B., York, J., "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing,* vol. 7, pp. 76-80, 2003.

[17] B. M. Sarwar, Karypis, G., Konstan, J.A., Riedl,J., "Application of dimensionality reduction in recommender system — a case study," presented at the WebKDD 2000, 2000.

[18] B. Mehta, Nejdl, W., "Unsupervised strategies for shilling detection and robust collaborative filtering.," *User Modeling and User-Adapted Interaction,* vol. 19, pp. 65-97, 2009.

[19] H. X. Pham, Jung, J.J., "Preference-based user rating correction process for interactive recommendation systems," *Multimedia Tools and Applications,* 2012.

[20] B. Li, Chen, L., Xingquan, Z., Chengqi, Z., "Noisy but non-malicious user detection in social recommender systems," *World Wide Web,* 2012.

[21] A. Gunawardana, Shani, G., "A survey of accuracy evaluation metrics of recommendation tasks," *Journal of Machine Learning Research,* vol. 10, pp. 2935-2962, 2009.