Knowledge-Based Systems 76 (2015) 96–108

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Correcting noisy ratings in collaborative recommender systems

Raciel Yera Toledo [a], Yailé Caballero Mota [b], Luis Martínez [c,*]

[a] Knowledge Management Center, University of Ciego de Ávila, Carretera a Morón Km 9½, C.A., Cuba
[b] Department of Computer Science, University of Camagüey, Circun. Km 5½, Camagüey, Cuba
[c] Department of Computer Science, University of Jaen, 23071 Jaen, Spain

ARTICLE INFO

Article history:
Received 10 January 2014
Received in revised form 8 November 2014
Accepted 7 December 2014
Available online 19 December 2014

Keywords:
Natural noise
Collaborative filtering
Recommender systems
Nearest neighbor-based recommendation
Matrix factorization

ABSTRACT

Recommender systems help users to find information that best fits their preferences and needs in an overloaded search space. Most recommender systems research has been focused on the accuracy improvement of recommendation algorithms. Despite this, recently new trends in recommender systems have become important research topics such as, cold start, group recommendations, context-aware recommendations, and natural noise. The concept of *natural noise* is related to the study and management of inconsistencies in datasets of users' preferences used in recommender systems. In this paper a novel approach is proposed to *detect* and *correct* those inconsistent ratings that might bias recommendations, whose main advantage regarding previous proposals is that it uses only the current ratings in the dataset without needing any additional information. To do so, this proposal detects noisy ratings by characterizing items and users by their profiles, and then a strategy to fix these noisy ratings is carried out to increase the accuracy of such recommender systems. Finally a case study is developed to show the advantage of this proposal to deal with natural noise regarding previous methodologies.

1. Introduction

Recommender systems have become a mainstream research field in information technologies. They appear as a supporting tool for helping users trying to obtain information that best fits their preferences and needs, in scenarios in which the information overload is an important drawback.

The most popular version of Recommender Systems (RS) [1], *learns* from user preferences about a predefined set of known items, and predicts the preference degree about unknown items. With this goal, many applications have been built to recommend different types of items like movies, books, TV shows, jokes, news, scientific papers, web pages, and so on. They have covered diverse areas like e-commerce [2], e-learning [3,4], e-services [5], tourism [6,7], and software engineering [8].

The two most used techniques in the development of RS are the content-based and the collaborative filtering ones. Content-based recommender systems [9,10], suggest items with similar features to those that the user chose in the past. On the other hand, collaborative filtering recommender systems [11,12] recommend items that other similar users liked in the past. Collaborative filtering systems can generate recommendations just using information about users' preferences regarding a set of items, and due to their simplicity they have become popular nowadays.

Most of research done in collaborative filtering has been focused on tuning basic algorithms to improve their accuracy [11–16]. It includes neighborhood-based user-user and item-item approaches, dimensionality reduction approaches, probabilistic, MDP-based, graph-based, rule-based methods, and their hybridization [17]. In past years, advances in collaborative filtering [18] have overcome important challenges like the Netflix Prize, showing that the rating prediction can be significantly improved.

Unlike classical data mining processes in which the data is usually considered to contain a degree of inconsistency, and therefore data preprocessing is an important step in the mining processes [19], in RS it is assumed that the ratings in the datasets are free of irregularities. Nevertheless, recently Amatriain et al. [20] have shown that users could be inconsistent when they elicit ratings for items, exposing recommender systems data to inconsistencies. These inconsistencies, so called *natural noise* [21], are inserted without malicious intent, and its manipulation represents an important open problem to improve the recommender systems performance.

Several approaches have been introduced to manage natural noise in recommender systems [11,22,23]. O'Mahony et al. introduced in [21] a proposal that ignores noisy ratings, whilst others

* Corresponding author.
E-mail addresses: ryera@cegic.unica.cu (R.Y. Toledo), yaile.caballero@reduc.edu.cu (Y.C. Mota), luis.martinez@ujaen.es (L. Martínez).

http://dx.doi.org/10.1016/j.knosys.2014.12.011
0950-7051/

correct them by requiring additional information like metadata and users' data [22,24].

In this paper a novel approach is introduced to detect and correct noisy ratings whose main novelty regarding previous approaches is that it just uses current ratings, i.e., *it does not require any additional information to manage the natural noise*. With this aim, our proposal is composed of two phases: it initially characterizes users and items according to their profiles and infers a noisy classification of ratings from this characterization, in order to identify a set of possibly noisy ratings. Once these ratings have been detected a fixing strategy is developed to correct them if necessary. To validate the hypothesis that the corrected data improve the accuracy of collaborative filtering recommender systems, a case study of two well-known datasets as Movielens and MovieTweeting is carried out.

The main contribution of this paper is then to present a natural noise correction framework that only exploit implicit knowledge behind the rating matrix, and does not depend on additional information. More specifically, our proposal consists of extracting ratings knowledge by means of user, item and rating classifications, and using them to find out noisy preferences. Once the noisy ratings have been detected, a new and simple but effective approach based on collaborative filtering prediction algorithms is carried out to correct them.

The paper is organized as follows: Section 2 reviews collaborative filtering and relevant research about natural noise and rating correction. Section 3 introduces a novel approach for detecting and managing natural noise without any additional data. Section 4 performs and discusses several experiments to evaluate the previous proposal in comparison with other approaches for managing natural noise in recommender systems. Section 5 provides an analysis of the results obtained, and finally, Section 6 concludes the contribution.

2. Background

In this section a brief review of collaborative filtering is introduced for a better understanding of our proposal and different related works about natural noise are briefly described to clarify the importance of this topic in recommender systems.

2.1. Collaborative filtering

Different algorithms have been used to develop Collaborative Filtering Recommender Systems (CFRS). Here, we review some of the most widely used approaches in the literature, as they will be used in our case study. Afterwards two evaluation methods for CFRS are introduced.

2.1.1. Collaborative filtering approaches

CFRS provide recommendations to users for items that people with similar tastes have already liked. While other recommender system approaches depend on content associated to items, the collaborative filtering approach generally uses only ratings to predict user's preferences for items that the user has not experienced yet [1,11,17,25].

In order to facilitate the evaluation of our proposal for managing natural noise in CFRS from the different collaborative filtering approaches we will use two classical and wide broad approaches [17]:

(a) *Item-based collaborative filtering* [26]. It predicts the user's preferences for items that have not been rated yet, looking for other items similar to the active one taking into account the similarities previously computed and stored in a model. It then uses the current user's ratings for those items to predict the unknown rating.

In our case study we will use an alternative proposed by Sarwar et al. [27] that is based on the use of Pearson's correlation coefficient to calculate the similarity between items (Eq. (1)), and a weighted sum of the neighbors' preferences to predict the required rating (Eq. (2)). In these equations, $r_{u,i}$ represents the rating provided by the user u for the item i, $\overline{r_i}$ and $\overline{r_j}$ represent the average rating for items i and j considering only the preferences of those users which have co-evaluated both items, and $w(i,j)$ represents the similarity value. Finally, P_{ui} represents the final prediction for the user u regarding the item i.

$$w(i,j) = \frac{\sum_{u\in U}(r_{u,i}-\overline{r_i})(r_{u,j}-\overline{r_j})}{\sqrt{\sum_{u\in U}(r_{u,i}-\overline{r_i})^2\sum_{u\in U}(r_{u,j}-\overline{r_j})^2}} \quad (1)$$

$$P_{u,i} = \frac{\sum_{j\in Neighbors} r_{u,j} w(i,j)}{\sum_{j\in Neighbors}|w(i,j)|} \quad (2)$$

(b) *Matrix factorization approach.* It appears as a solution for the classical scalability and sparsity problems in the more traditional collaborative filtering methods. Specifically, 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's preferences can be expressed considering user's interests in a topic, and the degree to which each item is significant to the topic. Early works like [28] formally use latent semantic analysis in the form of truncated singular value decomposition (SVD) to perform this dimensionality reduction. However, over the last few years less expensive and more accurate methods have been developed using alternative procedures with the same purpose. These proposals, summarized in [18], focus on learning a new space by using gradient descendent techniques, instead of transforming the initial rating matrix in a new one. This new space is then used to calculate the rating predictions.

In this direction, Koren et al. [18] propose associating each user u and item i with k-dimensional vectors p_u and q_i, and then representing a rating $r^*_{u,i}$ for a specific user and item, as a dot product between the corresponding vectors (Eq. (3)). Following this model, the rating values are used to *learn* the k-dimensional vectors, minimizing the regularized square error on the set of known ratings $r_{u,i}$ (Eq. (4)). In this equation, $r_{u,i}$ represents the original rating value, and $\lambda(||q_i||^2 + ||p_u||^2)$ the regularization term inserted to avoid overfitting. In this term, the regularization value λ is usually experimentally determined. Once vectors are obtained, they could be used to calculate the rating value for any user and item, using Eq. (3).

$$r^*_{u,i} = q_i^T p_u \quad (3)$$

$$\min_{q^*,p^*} \sum_{(u,i)\in R} (r_{u,i} - q_i^T p_u)^2 + \lambda\left(\|q_i\|^2 + \|p_u\|^2\right) \quad (4)$$

A popular extension of this framework, also presented by Koren et al. [18], considers the bias involved in the prediction calculated by the Equation (3). This bias includes an overall rating μ, and parameters b_u and b_i representing the corresponding deviation from the average for the user u and item i, associated to the corresponding rating. Eq. (5)

presents this new model, where the parameters to optimize (q^*, p^*, b^*) are determined through a similar method to that used in the Eq. (4). In this case, the regularization term is also updated in order to add the new parameters to optimize this scenario (b_u and b_i).

$$\min_{q^*,p^*,b^*} \sum_{(u,i)\in R} (r_{u,i} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2) \tag{5}$$

2.1.2. On the evaluation of collaborative recommender systems

Several frameworks have been proposed for offline and online evaluations of collaborative recommendation systems [11,12,29].

Gunawardana and Shani [30] specify a *protocol* to perform this evaluation task using a dataset with ratings associated to users and items. It proposes to select a set of users from the original dataset. Then, it randomly selects the amount of items n_u to hide for each user u. These hidden items will be the test set, and remaining ones the training set.

Using these training and test sets, the referred work proposes to compute a new prediction for each test rating, by using the prediction algorithm. These predictions are used to calculate the quality/performance of the algorithm by means of different metrics. The two more popular metrics are:

(a) The mean absolute error (MAE) for all the predictions made: MAE is defined as the mean absolute error between the true test ratings and the predicted ratings.
(b) The $F1$ metric as a combination of precision and recall: Precision and recall measure the degree to which the system presents relevant information by computing the portion of both preferred and recommended items, from the total number of preferred items and recommended items. To combine precision and recall, the $F1$ metric is defined as $F1 = 2 * \text{precision} * \text{recall}/(\text{precision} + \text{recall})$. To measure $F1$ using predictions, a relevance threshold is established, and an item is assumed relevant if it has been rated with a value equal or greater than this threshold. On the other hand, all ratings lower than this threshold are non-relevant. Therefore, a different $F1$ value is computed for each user, and finally averaged to obtain the global $F1$ value. This approach to calculate $F1$, has been used previously by several authors [31–33].

The aforementioned evaluation process with its partitions and evaluation metrics will be used in our experiments to evaluate the effect of our proposal in the performance of CFRS (Section 4).

2.2. Related works: Natural noise in collaborative filtering

Noise in recommender systems datasets has been grouped in two main categories [21]:

(1) *Malicious noise*, associated to noise intentionally introduced by an external agent to bias recommender results [34], and
(2) *Natural noise*, involuntarily introduced by users, and that could also affect the recommendation result [35].

While handling malicious noise is a well-developed research area, natural noise is a very recent topic of research and has received much less attention. The natural noise is produced by different sources: while the malicious noise is usually associated to user profiles that match certain patterns [34], the natural noise identification is more difficult because it tends to appear in several ways dissimilar to each other [35]. For these reasons, techniques for processing both types of noise must be different.

With regards to natural noise, several authors have suggested that a rating never should be appreciated as a ground-truth value, because the process of eliciting preferences is intrinsically noisy [20,24]. With this perspective, Amatriain et al. [20] and Pham and Jung [24] outlined two possible reasons for the appearance of natural noise in recommendation datasets: (1) the user preferences change over time and (2) the users inherent imprecision for eliciting ratings. According to Said et al. [23] and Kluver et al. [36], this second reason is caused by several factors like personal conditions, social influences, emotional states, contexts, or certain rating scales.

Different user studies have been developed to support these statements [20,23,37]. Specifically, Amatriain et al. [20] made some experiments to quantify the users' inconsistency degree by analyzing their ratings for an item in different time intervals that vary from one day to 15 days, and conclude that in all cases the users tend to be inconsistent. This result justifies the fact that the inconsistencies appear due to the preferences changing over time (associated to the inconsistencies in the long-term) and also due to the users' imprecision (associated to the inconsistencies in the short-term). Moreover, this work also shows that these inconsistencies can considerably affect the recommendation accuracy, and for this reason they should be treated as noise.

In order to deal with natural noise, O'Mahony et al. [21] present an approach that uses natural and malicious noise terms. It detects noisy ratings comparing each rating with a new value predicted for the corresponding user and item. These predictions are calculated by using a memory-based user-to-user collaborative filtering method and a set of *genuine* user profiles manually obtained. It then discards ratings whose differences exceed a threshold considering this comparison process.

An interactive method to eliminate noisy ratings, so-called item re-rating, was also proposed by Amatriain et al. in [22]. In this study, noisy ratings and noisy users are previously identified in online rating trials done by the authors as a part of their research. When any noisy rating is discovered, it is necessary to ask the user for a new rating. The authors consider that this re-rating process will remove the natural noise present. It affirms that denoising extreme ratings yields greater performance than denoising mild ratings. However, the mandatory user participation is an important limitation to this method that makes it hard to apply.

Recently, another important group of works explores data beyond ratings to correct erroneous rating values in recommender systems [24,38]. They use item attributes to learn a user preference model, marking a rating as incorrect if it belongs to the current user preference model, but it is less than a predefined threshold and less than the mean rating for this user. They also propose a way to correct ratings, defining some criteria to classify a user as expert, and propose three correction methods using experts. The first one considers a weighted average rating of experts for the same item. The second one considers a weighted average rating of items belonging to the user preference model (similarly to the item-to-item collaborative filtering), and the third one calculates the mean value of the previous two. This approach improves the recommender performance, but depends on item attributes to build the preference model. In many cases this information is not available and therefore these approaches cannot be applied.

Eventually, Li et al. [35] propose 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 values. For each user profile it quantifies underlying noise, obtaining those users with the higher noise degrees. In this work, the recommendation accuracy is improved when these users are removed from the dataset. However, this study focuses on noise detection on user level, so new approaches beyond users that deal with rating levels are needed. Noise treatment at this deeper level will facilitate the rat-

Table 1
Related work in natural noise treatment.

	Depends on additional information beyond ratings	Not depend on additional information beyond ratings
Removes noisy data		O'Mahony et al. [21] Li et al. [35]
Corrects noisy data	Amatriain et al. [22] Pham and Jung [24] Pham et al. [38]	

ing correction, avoiding the information removal from the dataset, demanded in [35].

Table 1 characterizes the different proposals to deal with natural noise according to the necessity of removing or fixing noisy data or the necessity of additional information beyond current ratings. Table 1 shows a shadowed cell that represents the lack of research centered on noisy ratings correction without additional information. Our proposal aims at filling this gap.

3. Managing natural noise by using ratings

Here, a novel approach is presented to detect and correct natural noisy ratings only taking into account rating values, without needing any additional information that was necessary in previous works [22,24,38]. With this aim, our proposal considers the fact, recently demonstrated by Hu and Pu [39] and Cantador et al. [40], that the users personality is related to the ratings they provide, and under the straightforward consequence that each item has a global preference degree according to the users' preferences. Specifically, it consists of a user classification and an item classification (just based on ratings), and assumes that for each rating, a contradiction between the corresponding user class, item class, and the value itself, represents evidence of possible noise. For this reason, our proposal adopts the fact that in a recommendation scenario where only preference values are available, the rating tendencies associated with users and items could be valuable information to detect noisy ratings.

In order to manage these natural noisy ratings, a framework is proposed that extends the traditional collaborative filtering paradigm (see Fig. 1). The basic problem associated with the collaborative filtering, represented with continuous lines, receives as inputs a set of past users' ratings and an unknown rating to determine for a certain user and item. Then, a prediction method (which is the core of the framework) provides an outcome for the requested preference. Represented by dotted lines, our proposal filters users' ratings out to detect and correct noisy ones, and hence provides free noisy ratings to the collaborative filtering method. Specifically, it includes two phases:

(1) *Noisy ratings detection:* it classifies user and item profiles and determines if the associated ratings are possibly noisy or not, taking into account this classification.

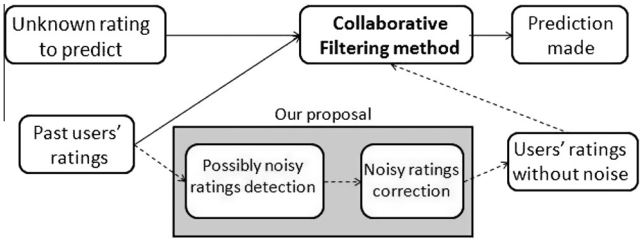

Fig. 1. The role of our proposal in a CF framework.

(2) *Noise correction:* it uses a basic collaborative filtering method to predict a new value for each possible noisy rating. This value will replace the noisy one if necessary.

A further detailed description of these two phases is introduced below:

3.1. Noisy rating detection

To detect natural noisy ratings, we initially perform a classification process for ratings, users, and items. We assume that each user has his/her own tendency when rating items, and that each item also has its own tendency receiving ratings. We identify four possible tendencies for users and items:

User tendencies:

(1) *Positive:* the user tends to evaluate all items positively.
(2) *Average:* the user provides average values.
(3) *Negative:* the user gives low ratings.
(4) *Hesitating:* the user oscillates between the previous categories, and does not fall into a specific one.

Item tendencies:

(1) *Preferred:* the item tends to be highly preferred by all users.
(2) *Av-Preferred:* the item tends to be averagely preferred.
(3) *No-Preferred:* the item is not preferred by most users.
(4) *Doubtful:* the item has contradictory opinions about its preference.

Eventually, we also classify each rating $r(u,i)$ (for a user u and an item i) into three different classes according its value:

Rating classes:

(1) *Weak:* if $r(u,i) < \kappa$.
(2) *Mean:* if $\kappa \leqslant r(u,i) < v$.
(3) *Strong:* when $r(u,i) \geqslant v$.

This classification depends on a weak-average threshold κ and an average-strong threshold v that must satisfy $\kappa < v$. These parameters could be based on global information or could be user dependent (and then referred as κ_u and v_u), or item dependent (κ_i and v_i). We will analyze their possible initial values later.

In order to facilitate the formal definition for each user and item tendencies, we propose grouping the ratings considering the mentioned classes and user and item dependent thresholds.

Since U and I are the sets of users and items, we group the preferences for each user u in the following sets W_u, A_u and S_u:

(1) Set of weak ratings provided for user u, W_u: $W_u = \{r(u,i) | \forall i \in I \text{ where } r(u,i) < \kappa_u\}$.
(2) Set of average ratings provided for user u, A_u: $A_u = \{r(u,i) | \forall i \in I \text{ where } \kappa_u \leqslant r(u,i) < v_u\}$.
(3) Set of strong ratings provided for user u, S_u: $S_u = \{r(u,i) | \forall i \in I \text{ where } r(u,i) \geqslant v_u\}$.

On the other hand, for each item I we group the preferences in the sets W_i, A_i and S_i:

(1) Set of weak ratings assigned to item i, W_i: $W_i = \{r(u,i) | \forall u \in U \text{ where } r(u,i) < \kappa_i\}$.
(2) Set of average ratings assigned to item i, A_i: $A_i = \{r(u,i) | \forall u \in U \text{ where } \kappa_i \leqslant r(u,i) < v_i\}$.
(3) Set of strong ratings assigned to item i, S_i: $S_i = \{r(u,i) | \forall u \in U \text{ where } r(u,i) \geqslant v_i\}$.

Considering these sets and the rating classes, we can formally define new classes associated for each user and item tendency previously mentioned. In the case of the users, we tag them as:

- *Benevolent:* users that belong to the *positive* group.
- *Average:* those ones that belong to the *average* group.
- *Critical:* those ones that belong to the *negative* group.
- *Variable: hesitating:* users.

In a similar way we define a classification for items, tagging them as:

- *Strongly-preferred:* items that belong to the *preferred* group.
- *Averagely-preferred:* those ones that belong to the *Av-preferred* group.
- *Weakly-preferred:* items that belong to the *No-preferred* group.
- *Variably-preferred:* those that belong to the *Doubtful* group.

Therefore, accomplishing such classifications is proposed by using the cardinality of the sets W_u, A_u, S_u, or W_i, A_i, S_i. Each user or item is classified depending on the cardinality associated with a set regarding the combined cardinality of the other two sets (see Table 2).

In Table 2, $|X|$ represents the cardinality of the set X.

Once each rating, user and item has been classified, we use this information to look for noisy ratings by analyzing the presence of contradictions among the classes. With this purpose, we define three groups of homologous classes respectively associated with users, items, and ratings (Table 3).

Specifically, the detection process assumes that for a rating $r(u,i)$ from the recommender dataset, if the classes associated with the user u and the item i belong to the same group in Table 3, then the rating $r(u,i)$ must belong to the rating class in the same group. If the rating does not verify this, it could be then a noisy rating and it is tagged as *possible noise*, and its transformation might mitigate the natural noise associated with the dataset.

This strategy is not applicable for variable user's ratings, neither for ratings associated with variably-preferred items, because there is no homologous rating class for these user and item classes.

Remark 1. It is important to highlight that user and item classification provides a *variable* class which means both users and items can present different type of behaviors over time. In such cases it makes no sense to look for noisy ratings.

Table 2
User and item classes.

User classes	
Critical user	$\|W_u\| \geqslant \|A_u\| + \|S_u\|$
Average user	$\|A_u\| \geqslant \|W_u\| + \|S_u\|$
Benevolent user	$\|S_u\| \geqslant \|W_u\| + \|A_u\|$
Variable user	Does not satisfy the other user conditions
Item classes	
Weakly-preferred item	$\|W_i\| \geqslant \|A_i\| + \|S_i\|$
Averagely-preferred item	$\|A_i\| \geqslant \|W_i\| + \|S_i\|$
Strongly-preferred item	$\|S_i\| \geqslant \|W_i\| + \|A_i\|$
Variably-preferred item	Does not satisfy the other item conditions

Table 3
Homologous classes.

	User class	Item class	Rating class
Group 1	*Critical*	*Weakly-preferred*	*Weak*
Group 2	*Average*	*Averagely-preferred*	*Average*
Group 3	*Benevolent*	*Strongly-preferred*	*Strong*

The Algorithm 1 summarizes the entire approach for detecting ratings that could be noisy.

Algorithm 1. Detection of possibly noisy ratings

Input: r = {$r(u,i)$**}** – set of available ratings, κ_u, ν_u, κ_i, ν_i, κ, ν, – classification thresholds
Output: possible_noise = {$r(u,i)$} – set of possible noisy ratings

```
W_u = {}, W_i = {}, A_u = {}, A_i = {}, S_u = {}, S_i = {}
possible_noise = {}
for each rating r(u,i)
    if r(u,i) < κ_u
        Add r(u,i) to the set W_u
    else if r(u,i) >= κ_u and r(u,i) < ν_u
        Add r(u,i) to the set A_u
    else
        Add r(u,i) to the set S_u
    if r(u,i) < κ_i
        Add r(u,i) to the set W_i
    else if r(u,i) >= κ_i and r(u,i) < ν_i
        Add r(u,i) to the set A_i
    else
        Add r(u,i) to the set S_i
end for
for each user u and item i
Classify it using the associated sets, according to the definitions in Table 2
end for
for each rating r(u,i)
    if u is critical, i is weakly-preferred, and r(u,i) >= κ
        Add r(u,i) to the set possible_noise
    if u is average, i is averagely-preferred, and (r(u,i) < κ or r(u,i) >= ν)
        Add r(u,i) to the set possible_noise
    if u is benevolent, i is strongly-preferred, and r(u,i) < ν
        Add r(u,i) to the set possible_noise
end for
```

3.2. Noise correction process

Once the method has detected the possibly noisy ratings, the next phase manages such ratings. In the literature there are different choices, either discard noisy ratings [21] or correct them [24]. Our proposal adopts the latter view, assuming that several authors suggest to correct the anomalous information from the data (if it is possible), instead of removing it in order to avoid loss of information [41]. Our strategy modifies possibly noisy ratings, r, by predicting a new rating r^* using a traditional collaborative filtering prediction algorithm with the original training set [11,12]. If $|r - r^*| > \delta$ then r is replaced by r^* otherwise r is kept, δ being a threshold.

The prediction algorithm selected for the correction process is a user-user collaborative filtering approach with Pearson's similarity and $k = 60$ neighbors. This method was pioneer in the collaborative filtering field and, does not need the construction of intermediate knowledge. This feature alleviates the computational cost.

Algorithm 2 presents an overview of the correction process. It initially invokes the Algorithm 1 and then iterates over the possibly noisy ratings to perform the correction based on the new rating prediction using the entire ratings set.

Algorithm 2. Noisy ratings correction

Input: $\mathbf{r} = \{r(u,i)\}$ – set of available ratings, δ – difference threshold
Output: $\mathbf{r}^* = \{r(u,i)\}$ – set of available ratings corrected
01 poss_noise = possible_noise_detection()
02 for each rating $r(u,i)$ in poss_noise
03 Predict a new rating $n(u,i)$ for user u and item i, using user–user memory-based collaborative filtering with Pearson's correlation coefficient, and using r as the training set
04 if $(\text{abs}(n(u,i) - r(u,i)) > \delta)$
05 Replace $r(u,i)$ by $n(u,i)$ in the original rating set r
06 end for

3.3. On parameter values

The proposal introduced previously depends on three parameter groups: the weak-average thresholds (κ_u, κ_i, and κ), the average-strong thresholds (v_u, v_u, and v), and the difference threshold (δ). These parameters are highly domain-dependent, so it is difficult to predetermine their optimal values. However, a strategy could be defined to assign them good initial values.

Therefore, we plan to evaluate two approaches to initialize them. The first one follows a global perspective, while the second one tries to adapt the initial values according to the corresponding user and item.

3.3.1. Global perspective

Due to the fact that the ratings are initially ordinal values on a scale and we assumed three possible classes for the ratings, the values for the thresholds (κ and v) should be selected in a way that approximately divides the rating range into three equal bins.

The Eqs. (6) and (7) calculate the values for all κ and v following these criteria, considering the use of $round(n)$ as a function to determine the nearest entire value for n, and also using the minimum and maximum possible values for the ratings.

$$\kappa = \kappa_u = \kappa_i = \min R + round\left(\frac{1}{3} * (\max R - \min R)\right) \tag{6}$$

$$v = v_u = v_i = \max R - round\left(\frac{1}{3} * (\max R - \min R)\right) \tag{7}$$

On the other hand, considering that the rating values are represented on a scale, we suggest assigning the value of a minimum step on 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 on the rating scale, then it is too large to replace it.

We will refer to this approach as the *global-pv* approach for the parameter values.

Remark 2. Different studies about the parameter values on several datasets have been done, their findings concluded that the suggested method provides the best values or very close to them.

3.3.2. Adapting initial values

Here the values are adapted according to the rating distribution for the corresponding users and items. Specifically, we follow the principle that maybe most of the ratings for certain users and items could be located in a specific section of the rating range, and still should be associated to different classes. To model this fact, we focused on the mean x' and the standard deviation p' for each user u and item i, and respectively defined them as x'_u, p'_u, x'_i, and p'_i. We then defined the thresholds κ_u, v_u, κ_i, v_i as follow in equations (8)–(11).

$$\kappa_u = x'_u - p'_u \tag{8}$$

$$v_u = x'_u + p'_u \tag{9}$$

$$\kappa_i = x'_i - p'_i \tag{10}$$

$$v_i = x'_i + p'_i \tag{11}$$

To calculate the values for κ and v, we will consider two alternatives: to perform it based on users ($\kappa = \kappa_u$ and $v = v_u$), or to perform it based on items ($\kappa = \kappa_i$ and $v = v_i$).

Finally, related to the difference threshold (δ), in this case we also use the mentioned statistical measures and then also regard the two alternatives: $\delta = p'_u$ (based on users), and $\delta = p'_i$ (based on items).

These approaches are referred as *user-based-pv* ($\kappa = \kappa_u$, $v = v_u$, $\delta = p'_u$) and *item-based-pv* ($\kappa = \kappa_i$, $v = v_i$, $\delta = p'_i$).

3.4. Illustrative example

In this section, we will provide a simplified recommendation scenario of use to show clearly how the proposal could detect and fix natural noisy ratings in a CFRS.

Table 4 presents a rating matrix in the range [1,5], with four users and four items. For these data and with demonstrating purposes, we initially apply the Algorithm 1, being $\kappa = 2$, $v = 4$, $\delta = 1$ (the global-pv alternative) and calculating the sets W_u, A_u, S_u, W_i, A_i, and S_i by using their cardinality to perform the classification. Table 5 presents the results obtained in this step, in which there are three benevolent users and one variable user, three strongly-preferred items and one averagely-preferred item.

The final step is then applied in Algorithm 1, to find contradictions. As a result, the exploration discovers the rating r_{u4i1} as possibly noisy, because although the associated user $u4$ is benevolent and the item $i1$ is strongly-preferred, the current rating is not strong.

Finally, Algorithm 2 verifies the necessity of correcting the possibly noisy ratings. First, it predicts a new rating value for the user $u4$ and the item $i1$, by using the entire dataset and the memory-based collaborative filtering method already specified (in this case with $k = 2$). The new rating value is $r_{u4i1} = 3.05$, and given that the difference between the old and the new value is greater than one, we then perform the rating replacement in the original matrix.

The scenario is a simple illustrative example of how our approach carries out the rating correction for CFRS. In the next section how these transformations have a positive impact on the recommendation accuracy is shown.

4. Experiments and results

This section presents the impact of our proposal as a preprocessing step in the recommendation accuracy associated with the CFRS approaches reviewed in Section 2. Additionally,

Table 4
Scenario of use for the proposal.

	$i1$	$i2$	$i3$	$i4$
$u1$	5	4	3	4
$u2$	5	4	1	4
$u3$	5		3	1
$u4$	2	5	5	5

Table 5
Users and items classification for the ratings in Table 4.

Users classification				
	$\|W_u\|$	$\|A_u\|$	$\|S_u\|$	Class
$u1$	0	1	3	Benevolent
$u2$	1	0	3	Benevolent
$u3$	1	1	1	Variable
$u4$	0	1	3	Benevolent
Items classification				
	$\|W_i\|$	$\|A_i\|$	$\|S_i\|$	Class
$i1$	0	1	3	Strongly-pref.
$i2$	0	0	3	Strongly-pref.
$i3$	1	2	1	Averagely-pref.
$i4$	1	0	3	Strongly-pref.

the evaluation includes a comparison regarding the benchmark methods. Finally, two critical issues also associated with this kind of proposal are referred to: the analysis of the method's intrusiveness and the analysis of its computational performance.

4.1. Datasets

Our study uses two well-known datasets in recommendation systems research. First, is Movielens, that contains 100,000 movie ratings on 943 users and 1682 items where each rating is in the interval [1,5]. Second, is MovieTweeting [42], with around 140,000 movie ratings provided by 21,018 users about 12,569 items in November 2013, and in this case the ratings are in the interval [0,10].

To prepare the data for the experiments, we randomly select 900 users in the case of Movielens, and all the users with more than 20 ratings in the case of MovieTweeting. These users were used to build the training and the test set as we indicated in Section 2.1.2.

To measure the effects of our correction process, we compare the performance of the collaborative filtering algorithm predicting the test set ratings using the transformed training set, and the performance using the original training set. This performance is calculated by using the metrics MAE and $F1$. Specifically, we obtain the $F1$ values considering relevant those rating values that belong to the higher quarter of the rating scale. We then use a relevance threshold = 4 for Movielens (with a scale [1,5]), and a relevance threshold = 7.5 for MovieTweeting (with a scale [0,10]).

4.2. Parameter setup

In order to measure the behavior of the two CF methods reviewed in Section 2 with and without the rating correction provided by our proposal, the basic CF implementations provided by MyMediaLite v1.0 will be used [43]. In the case of the matrix factorization-based method, it allows the specification of two different regularization extent values (λ) for the vectors and for the biases. Considering other aspects and according to Gantner et al. [43], MyMediaLite exactly implements the two methods mentioned in Section 2.

With this aim, the current experimental scenario has two groups of parameters: those associated with the traditional CF approaches, and those associated with the proposal itself.

In the first case and assuming that our goal is to determine the effect of the correction in the recommendation accuracy, we globally define some parameter values in the biased matrix factorization approach for all the experiments. Those parameters were the vector regularization value ($\lambda_1 = 0.015$), the biases regularization value ($\lambda_2 = 0.015$) and the learning rate value ($\alpha = 0.01$). We appreciate that different values on these parameters do not directly affect our comparison purpose.

For other parameters like the number of vectors associated to the model (num_factors) and the number of iterations used to load the model (num_iters), we empirically predefine their values for each dataset in order to guarantee the best corresponding performance of the method. Then, in both cases we set up num_factors = 5 and num_iters = 10.

In the case of the item-based collaborative filtering method, we select $k = 60$ for the amount of nearest neighbors to use. This value tends to be used to evaluate this kind of methods in several scenarios [33].

In relation to the parameter values for the proposal itself, for the *global-pv* approach (Section 3.3), in Movielens we set the values $\kappa = 2$ for weak-average threshold and $\nu = 4$ for average-strong threshold, and in MovieTweeting we set the values $\kappa = 3$ and $\nu = 7$. In the case of the difference threshold between the predicted and the original rating, in both scenarios we use the minimum difference between the original ratings in their scales, and then assign $\delta = 1$.

Finally, the *user-based-pv* and *item-based-pv* approaches assign the personalized values specified in Section 3.3.

4.3. Benchmark methods

Section 2.2 has presented some previous works focused on handling natural noise in collaborative filtering. Specifically, we select the methods proposed by O'Mahony et al. [21] and Li et al. [35] to compare against our proposal, provided that the remaining methods depend on additional information that is not currently available. Additionally, our proposal is also compared with three approaches that remove malicious noise [44,45], to verify if they are also competitive improving the recommendation accuracy in the current scenario. Previous works like [35] have performed the comparison with these latter methods with the same purposes.

For these comparisons, we classify the five methods in two new groups considering how noise is removed in the dataset:

(i) O'Mahony et al. [21] focus on the detection and removal of noisy ratings. This method (Section 2.2) removes the noisy ratings in the training set, and evaluates the performance in the test set using the modified training set.

(ii) The remaining four approaches proposed by Li et al. [35] (NNMU), Chirita et al. [44] and Burke et al. [45] (RDMA, WDMA and WDA respectively) perform the removal on the complete user profiles providing an ordered user ranking considering their noise degree. Hence these methods select the top k noisiest users, remove their ratings for the training set, and evaluate the recommendation accuracy by using the remaining data in the training set, and the original test set. Therefore, the predictions for the ratings in the test set associated to users deleted in the training set, are done by returning the global rating bias [46].

4.4. Evaluation results

This section presents the results associated with the effect of our proposal on the CFRS presented in Section 2, and its comparison regarding the benchmark approaches.

4.4.1. The effect of our proposal in CFRS methods

Table 6 shows the performance of our method according to the alternatives for initializing the parameters (*global-pv, user-based-pv*, and *item-based-pv*). For all cases, at least one of these alternatives outperforms the baseline algorithm. We remark that still in a dimensionality reduction method like the matrix factorization

Table 6
MAE values with and without rating correction in Movielens and MovieTweeting (item-based and matrix factorization-based collaborative filtering). Bolded values mean best results.

Item-based CF				
	Baseline	PrevClassBased + global-pv	PrevClassBased + user-based-pv	PrevClassBased + item-based-pv
MAE Movielens	0.7705	0.7673	**0.7659**	0.7678
MAE Movie-Tweeting	1.2077	**1.1860**	1.1920	1.1860
F1 metric Movielens	0.4650	**0.5103**	0.4816	0.4934
F1 metric Movie-Tweeting	0.4944	**0.5278**	0.4744	0.4972
MF-based CF				
	Baseline	PrevClassBased + global-pv	PrevClassBased + user-based-pv	PrevClassBased + item-based-pv
MAE Movielens	0.7625	**0.7608**	0.7624	0.7625
MAE Movie-Tweeting	1.1739	1.1653	1.1721	**1.1582**
F1 metric Movielens	0.4344	**0.5037**	0.4684	0.4702
F1 metric Movie-Tweeting	0.5225	**0.5422**	0.4972	0.5153

approach that implicitly decreases the impact of noise by removing small disturbances [11], our framework provides an accuracy improvement.

However, it is worthy to note that the results suggest that the use of a more sophisticated approach (like user-based-pv or item-based-pv), does not necessarily imply a better performance. In fact, these two alternatives just improve global-pv for two specific datasets and recommendation method pairs (Movielens with the Item-Based CF and MovieTweeting with the MF-Based CF, both with the MAE metric). For the remaining cases, the global-pv approach improves the more complex of the two. For these reasons, we will take this approach as the reference for the rest of the paper, referring it as PrevClassBased for all cases.

Using just the *global-pv* alternative, Tables 7 and 8 present the data of users and items belonging to each defined class, for each dataset. In both cases, we observe that an important quantity of users and items belong to the benevolent and strongly-preferred classes, and few users to a critical behavior. In addition, it also indicates that the threshold δ has an important role in deciding which ratings will be finally modified, considering that several ratings tagged as possible noise after the classification step, are not finally corrected.

4.4.2. Comparison with other noise-treatment approaches in collaborative filtering

Table 9 presents the comparison between PrevClassBased (global-pv) and the approach proposed by O'Mahony et al. [21]. In this method, we set the parameter $\delta = 1$ for both datasets. In addition we select the entire set of users, as the set of the genuine user profiles that the method needs. Analyzing the results, *PrevClassBased* outperforms O'Mahony approach always. Fig. 2 shows the comparison between the number of modifications in the dataset and it is clear that our approach undoubtedly needs much fewer modifications to improve the accuracy.

On the other hand, Figs. 3 and 4 show the comparison results between PrevClassBased and the methods based on the removal of whole k user profiles. In this case, we evaluate each method modifying the parameter k in the range [10,60] with step 10, and register the best result for each case. We use these values to compare with the results obtained by PrevClassBased.

In the figures, each method is represented in the bars with the format *method-k* (RDMA-60, WDA-10, and so on), where k is the value associated with the higher accuracy for each method. The results indicate that methods focused on handling malicious noise

Table 7
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
4	184	577	135
Items per class			
Weakly-preferred item	Averagely-preferred item	Strongly-preferred item	Variably-preferred item
84	568	592	334
Amount of possible noisy ratings after classification		Amount of finally corrected ratings	
5404 (11.05% of ratings in training set)		2262 (4.62%)	

Table 8
Amount of users and items per class, ratings with possible noise, and ratings finally corrected in MovieTweeting.

Users per class			
Critical users	Average users	Benevolent users	Variable users
1	179	1301	210
Items per class			
Weakly-preferred item	Averagely-preferred item	Strongly-preferred item	Variably-preferred item
338	1862	4324	720
Amount of possible noisy ratings after classification		Amount of finally corrected ratings	
2761 (7.45% of ratings in training set)		1844 (4.97%)	

Table 9
Comparison between our approach and O'mahony et al. [21]. Bolded values mean best results.

	Item-Based CF + O'Mahony et al. [21]	Item-Based CF + PrevClassBased	MF-based CF + O'Mahony et al. [21]	MF-based CF + PrevClassBased
MAE Movielens	0.7712	**0.7673**	0.7693	**0.7608**
MAE Movie-Tweeting	1.2043	**1.1860**	1.1965	**1.1653**
F1 metric Movielens	0.5054	**0.5103**	0.5012	**0.5037**
F1 metric Movie-Tweeting	0.4934	**0.5278**	0.4791	**0.5422**

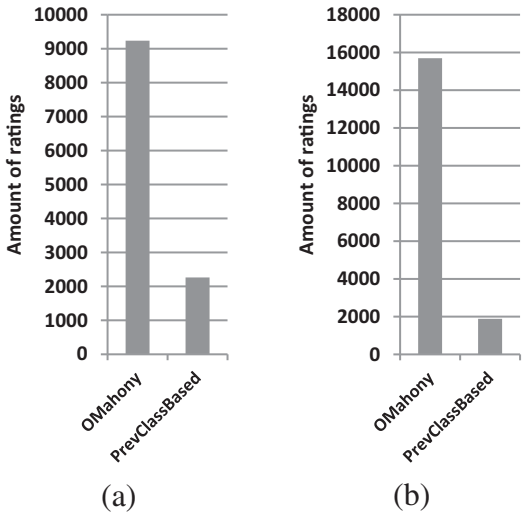

Fig. 2. Amount of ratings modified or removed for each case. (a) Movielens. (b) MovieTweeting.

have an even worse performance than the baseline, and their accuracy tends to decrease for higher values of *k*.

In contrast, NNMU outperforms the baseline considering MAE for MovieTweeting and for Movielens with the Item-Based CF method, but does not perform well according to the *F*1 metric. Beyond these findings, for all cases *PrevClassBased* notably outperforms all these approaches.

4.5. On the method's intrusiveness and its computational cost

Eventually, this section discusses two important facts in order to evaluate the proposal feasibility in real recommendation scenarios. The first discusses how intrusive it is in the user and item profiles modification, and the second is related to its computational performance.

4.5.1. Intrusiveness

To evaluate the intrusiveness of the method we have computed the amount of finally corrected ratings for each case, and the quantities were notably low. Specifically for Movielens, 85, 33% of users and 87, 15% of items required to modify less than 10% of their ratings. At MovieTweeting, the same percentages were calculated and the results were 80, 30% for users and 93, 15% for items. This concludes that the proposal only needs to modify a small portion of user and item profiles, and that it is a desirable feature in order to avoid a highly intrusive behavior.

4.5.2. Time cost

Regarding the time cost we can implement the proposal into two different scenarios:

1. *A preprocessing step executed on the system dataset to transform noisy ratings*: in this scenario the time cost will not affect the recommendations because the preprocessing process can be carried out offline.
2. *A previous step for each prediction generation in which the rating transformation would be temporally made just for the current prediction and then the original data would not be modified*: the detection of possibly noisy ratings could be done in just two passes over the rating matrix: the first one to perform the users and items classification, and the second one to discriminate the possibly noisy ratings based on the previous classification. Therefore, the correction of these ratings uses a prediction method that demands the similarity calculation for each pair of users in the data. For each rating modification, only the similarity values between the current user and those users that also evaluated the current item have to be modified in the correlation updating. An efficient implementation for the proposal could initially do the costly work by calculating each similarity value, and then for each rating correction just updates the similarities affected by this modification. The high sparsity associated with all recommender system dataset implies that these updates should be done quickly. Therefore, the computational cost of the method is just related to the cost of the whole user-user similarity calculation.

In order to verify this assumption we computed the execution time for each step of our method (Tables 10 and 11), and additionally compared it with the proposed by O'Mahony et al. [21]. With this purpose, we use a hardware platform composed by an Intel Pentium (R) Dual Core 3.06 GHz processor. We did not compare with Li et al. [35] because it has a notably longer response time. Regarding the comparison of both methods, the tables show that the possible noise detection step runs instantaneously, and also imply that the correction process runs faster, due to a notably fewer amount of ratings to transform (see Fig. 2). Additionally, the user-user similarity calculation was the most expensive task. Nevertheless, in our work we performed a pretty basic implementation for this procedure. Beyond the experimental environment, several authors have exposed different alternatives to accelerate this common step that could reduce its response time [47,48].

For these reasons, the computational cost should not be a problem for the present proposal, and allows its indistinct use for the two practical scenarios previously mentioned. Moreover, we think that specifically the possible noisy ratings detection allows its use as a previous step for each rating prediction (second scenario), in contrast to the work by O'Mahony et al. [21] that spent highly-valuable time on this context by correcting a higher amount of ratings.

5. Analysis of results

This section performs a complementary discussion about the experimental findings, and addresses potential future research actions. First, it is centered on summarizing the comparison section and justifies certain unexpected results, specifically related to the *F*1 values. Second, it is discussed reasons to justify the

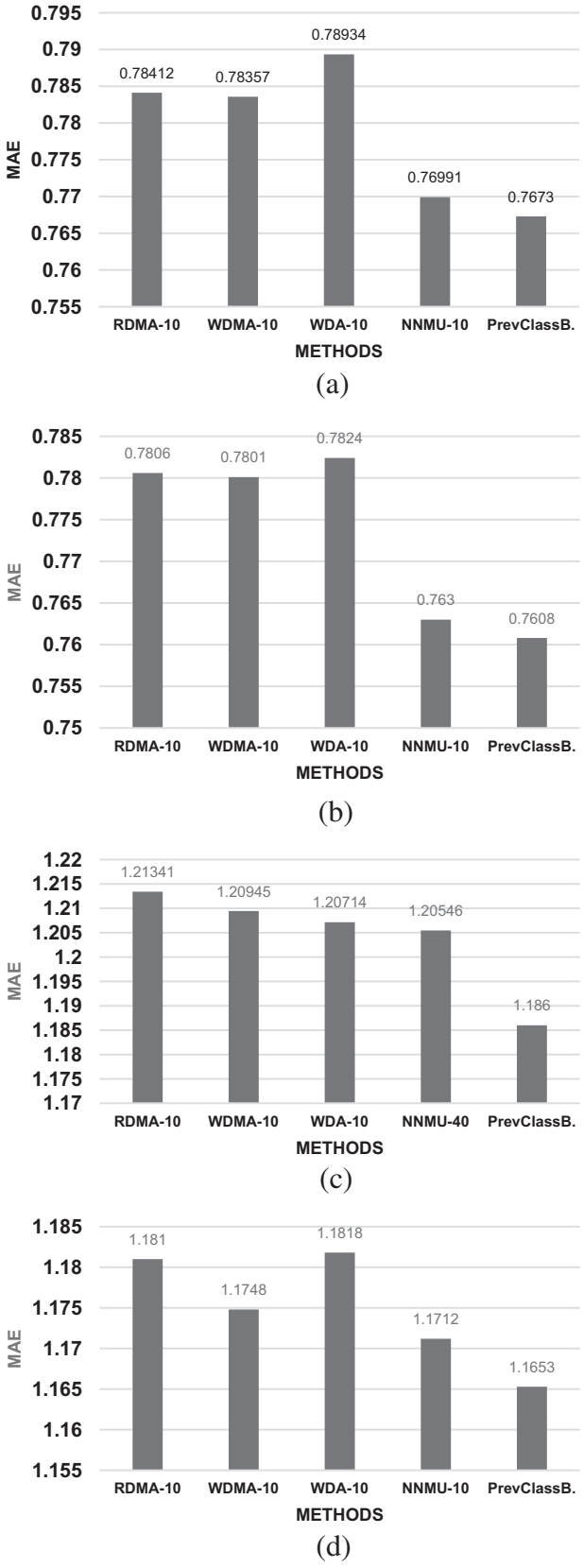

Fig. 3. Comparison against the methods based on profile removals, using MAE. (a) ItemKNN + Movielens. (b) MF + Movielens. (c) ItemKNN + MovieTweeting. (d) MF + MovieTweeting.

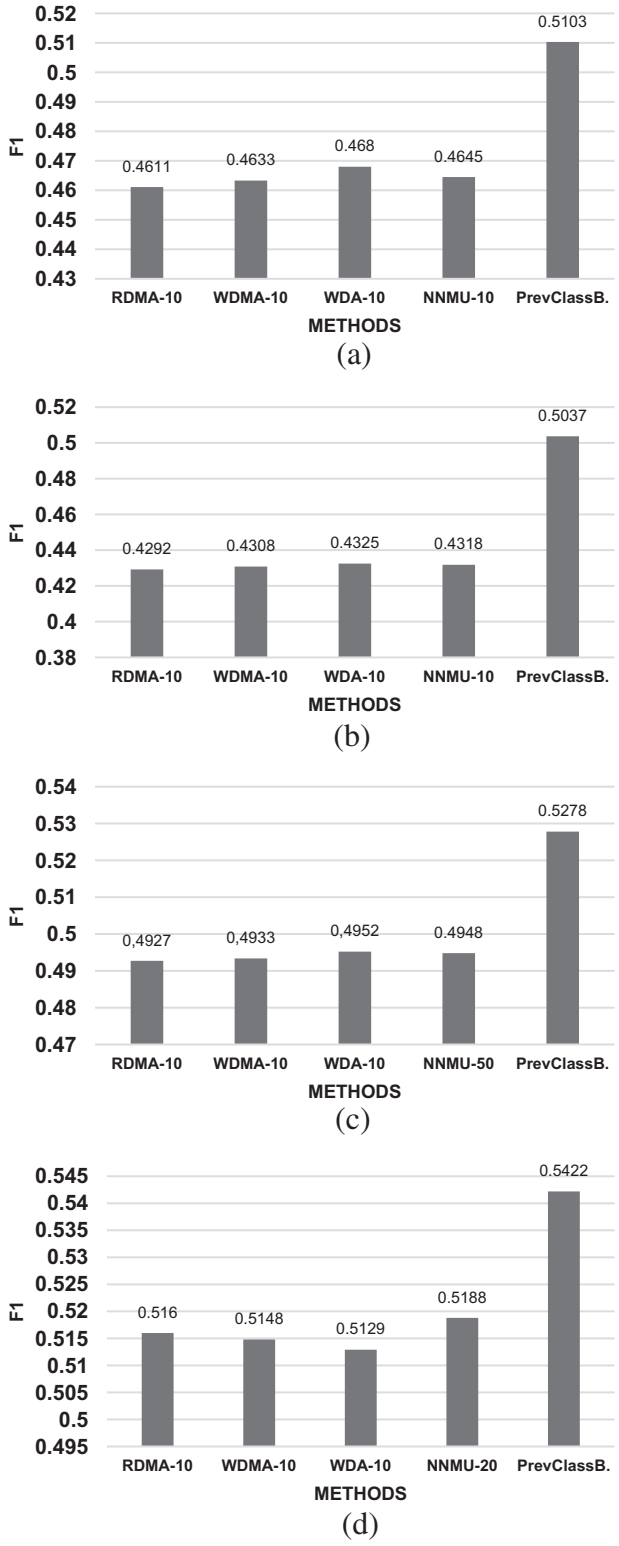

Fig. 4. Comparison against the methods based on profile removals, using F1. (a) ItemKNN + Movielens. (b) MF + Movielens. (c) ItemKNN + MovieTweeting. (d) MF + MovieTweeting.

feasibility of the noise correction only using ratings. Finally, we will point out some future directions for this research.

5.1. Regarding unexpected results

The results obtained in Section 4 show that the methods for managing *malicious noise* tend to underperform the baseline algorithm. On the other hand, the methods for managing *natural noise*

Table 10
Response time (in ms) for our approach and O'mahony et al. [21] in Movielens.

	Possible noise detection	Correlation calculation	Rating correction	Total
PrevClassBased	15	1406	1485	2906
O'Mahony et al. [21]	–	1406	8047	9453

Table 11
Response time (in ms) for our approach and O'mahony et al. [21] in MovieTweeting.

	Possible noise detection	Correlation calculation	Rating correction	Total
PrevClassBased	16	2906	578	3500
O'Mahony et al. [21]	–	2906	5625	8531

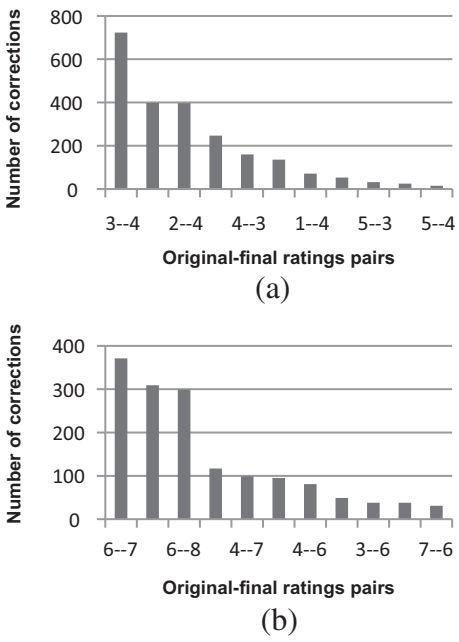

Fig. 5. Number of corrections grouped by the original rating value (at first), and the final value (at second). The groups were descendingly ordered by their size, and the graphics just present the biggest ones. (a) Movielens. (b) MovieTweeting.

slightly outperform the baseline with the exception of our proposal.

It is worthy to note the remarkable improvement associated with the $F1$ values between our proposal, and the baseline and the methods based on the removal of the complete user profiles. In order to find an explanation for this performance, we did a supplementary analysis that grouped, for each dataset, the transformed ratings concerning their original value and their final value after our correction process (rounded to the closest entire value). The results of this analysis are presented in Fig. 5. They indicate that most of the corrections increase the original rating values. Then, the transformation of some rating values into higher ones should imply that the predictions also tend to generate more elevated ratings; this fact combined with the empirically obtained conclusion (at least for the movie domains) that the ratings tend to have elevated values (and the test set should satisfy this feature), justifies the notable improvement on $F1$ taking into account this metric definition.

Additionally we would also like to observe that Fig. 5 shows that the ratings in the middle of the scale are more prone to be corrected than the extreme ratings. This conclusion was also obtained by Amatriain et al. [20], and it indicates that our work is aligned with similar researches in the same direction.

5.2. On the rating correction without additional information

Our two-step based proposal to remove the natural noise in ratings outperforms previous works in the sense that it is based on global knowledge extracted from the crowd preferences, instead of depending on additional information like user's and item's attributes. We think that this knowledge could be more valuable than the supplementary information. In this section, we present three additional reasons to justify the feasibility of noise correction only using ratings, and to suggest that the additional information is not a reliable source to be used profusely in this task.

1. Park et al. [49] pointed out that regarding the task of generating artificial user profiles (filterbots) to alleviate the cold start problem, the profiles generated by using global information mainly based on ratings, tend to achieve better results than those created through more attribute aware approaches. These latter profiles had a tendency to overfit contents too much.
2. To verify the initial statements, we focused on developing a parallel analysis to determine the actual context impact on rating distribution and consequently its possible impact on natural noise preprocessing. Specifically, we try to determine if we certainly could affirm that users love or hate a group of films regarding some attributes. With this purpose, we use the information related to the movie's genre also available in the Movielens original dataset. For each genre we calculate the ratio between the amount of preferred movies (four or five as rating value) and the non-preferred ones, regarding each user. The results obtained an average ratio under the initially expected value, where the highly popular genres (Action, Sci-Fi, Comedy, Adventure) were among the categories with lower values (always less than 3). This finding indicates that the preferences and dislikes of users regarding films associated with the same genre tend to be balanced. Moreover, it suggests that the preferences over a certain genre (as a context dimension), are not an appropriated attribute to characterize users, and then it do not provide reliable information to be used in a task like natural noise correction. We consider that this fact is also applicable to other attributes beyond genre.
3. Unlike the previous point, we think that there are some possibly implicit contexts that could represent the key for the natural noise treatment. However, the explicit discovery of those contexts is very difficult. Nevertheless, we consider that our rating correction approach implicitly takes into account these hidden contexts. In this case, the collaborative filtering method used to calculate the new rating value, only considers information about users located in the current user neighborhood. These users have similar preferences in relation to the current user, and for this reason represent the best information source to finally determine if a certain preference is noisy or not, beyond any additional dimension that might be used with this purpose.

We think that the results recently presented, which indicate that a common additional dimension like the genre has a low discriminative power to identify users, reinforce this fact.

Considering Table 4, in this case the correction of the rating for $u4$ was influenced by a neighborhood composed by users that are not actually similar to the current user. However, this example is actually a simplified scenario just conceived to show the method application. We consider that for a real scenario with a notably higher dimension, each user will have a high correlation with its neighborhood, avoiding the correction based on significantly different profiles that belong to different categories in a possibly hidden context.

5.3. Future research directions

The results obtained in the previous experimental sections indicate that it is possible to improve the accuracy of recommendation methods, through the detection and correction of inconsistent ratings without additional data. Tables 7 and 8 verify the hypothesis that the rating values allow us to classify users and items. However, we obtained a small number of critical users for both cases. This finding is closely related to the fact, empirically associated to the movie domain, that the users focus their preferences on the higher levels of the rating scales [32]. Therefore, we identify it as a weak point in our approach. We think that additional works to find more critical users could imply an accuracy improvement of the results currently achieved, regarding that Moon et al. [50] recently showed that users (specifically for the movie domain) could become critical under certain circumstances.

On the other hand, we have also identified the discarding of variable users and items for noise correction as a drawback, in the sense that they could also have inconsistent behaviors. Hence, further works must consider the possibilities of *softening* the restrictions presented in Table 2, in order to incorporate some variable users or variably-preferred items to other remaining categories.

6. Conclusions

This contribution provides a novel approach to deal with *natural noise* in Recommender Systems in order to improve recommendations. Specifically, it proposes an approach to manage 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 outperform the previous ones, and show the importance of managing natural noise to improve Recommender Systems performance.

This approach does not need any additional information to improve recommendations, because it is based on a user and item characterization that just uses the ratings in the current dataset. This is an important advantage regarding previous correction approaches in this area, that depend on supplementary information.

Our future research will be focused on providing new capabilities to our approach in order to detect and correct noisy ratings currently excluded like variable users and items. Specifically, we aim to manage the uncertainty in the users' and items' profiles, as a way to alleviate the restrictions that currently discard them.

Finally, we also want to extend the current study to other scenarios beyond the movies domain, to verify if the obtained results are consistent for other domains. At present, we have obtained preliminary results for a jokes domain (with the Jester dataset [51]) and for a books domain (with the BookCrossing dataset [52]), and for both cases, the class distribution was similar to that presented in Tables 7 and 8 for Movielens and MovieTweeting. Additionally, the recommendation accuracy was improved after the correction process. However, more experiments are required to establish similarities and some possible particularities for each case.

Acknowledgment

This work is partially supported by the Research Project TIN2012-31263 and FEDER funds. This work is funded by the eureka SD project (agreement number 2013-2591), that is supported by the Erasmus Mundus programme of the European Union.

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (2005) 734–749.
[2] J.B. Schafer, J.A. Konstan, J. Riedl, E-commerce recommendation applications, Data Min. Knowl. Disc. 5 (2001) 115–153.
[3] E.J. Castellano, L. Martínez, A web-decision support system based on collaborative filtering for academic orientation. Case study of the Spanish secondary school, J. Univ. Comp. Sci. 15 (2009) 2786–2807.
[4] J. Bobadilla, F. Serradilla, A. Hernando, Movielens, Collaborative filtering adapted to recommender systems of e-learning, Knowl.-Based Syst. 22 (2009) 261–265.
[5] Z. Zhang, H. Lin, K. Liu, W. Dianshuang, Z. Guangquan, J. Lu, A hybrid fuzzy-based personalized recommender system for telecom products/services, Inf. Sci. 235 (2013) 117–129.
[6] F. Ricci, Travel recommender systems, IEEE Intell. Syst. 17 (2002) 55–57.
[7] J.M. Noguera, M.J. Barranco, R.J. Segura, L. Martínez, A mobile 3D-GIS hybrid recommender system for tourism, Inf. Sci. 215 (2012) 37–52.
[8] H. Yang, C. Wang, Recommender system for software project planning: one application of revised CBR algorithm, Expert Syst. Appl. 36 (2009) 8938–8945.
[9] P. Lops, M. De Gemmis, G. Semeraro, Content-based recommender systems: state of the art and trends, in: L.R.F. Ricci, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2011, pp. 73–105.
[10] L. Martínez, L.G. Pérez, M.J. Barranco, A multigranular linguistic content-based recommendation model, Int. J. Intell. Syst. 22 (2007) 419–434.
[11] M.D. Ekstrand, J.T. Riedl, J.A. Konstan, Collaborative filtering recommender systems, Found. Trends Hum.–Comp. Interact. 4 (2010) 81–173.
[12] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, Recommender Systems Handbook, Springer Science + Business Media, 2011.
[13] M. Komkhao, J. Lu, Z. Li, W.A. Halang, Incremental collaborative filtering based on Mahalanobis distance and fuzzy membership for recommender systems, Int. J. Gen. Syst. 42 (2013) 41–66.
[14] X. Luo, Y. Xia, Z. Qingsheng, Y. Li, Boosting the K-nearest-neighborhood based incremental collaborative filtering, Knowl.-Based Syst. 53 (2013) 90–99.
[15] K. Choi et al., A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis, Electron. Commer. Res. Appl. 11 (2012) 309–317.
[16] H.-N. Kim et al., Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation, Electron. Comm. Res. Appl. 9 (2010) 73–83.
[17] X. Su, T. Khoshgoftaar, A survey of collaborative filtering techniques, Advan. Artif. Intell. 2009 (2009) 19.
[18] Y. Koren, R.M. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Comput. 42 (2009) 30–37.
[19] J. Han, M. Kamber, Data Mining: Concepts and Techniques, second ed. San Francisco, 2006.
[20] X. Amatriain, J. Pujol, N. Oliver, I like it... I like it not: evaluating user ratings noise in recommender systems, in: 17th International Conference on User Modeling, Adaptation and Personalization (UMAP), 2009, pp. 247–258.
[21] M.P. O'Mahony, N.J. Hurley, G.C. Silvestre, Detecting noise in recommender system databases, in: 11th ACM International Conference on Intelligent Users Interfaces (IUI), 2006, pp. 109–115.
[22] X. Amatriain, J. Pujol, N. Tintarev, N. Oliver, Rate it again: increasing recommendation accuracy by user re-rating, in: 3rd ACM International Conference on Recommender Systems (RecSys), 2009, pp. 173–180.
[23] A. Said, B.J. Jain, S. Narr, T. Plumbaum, Users and noise: the magic barrier of recommender systems, in: 23th International Conference on User Modeling, Adaptation and Personalization (UMAP '12), 2012, pp. 237–248.
[24] H.X. Pham, J.J. Jung, Preference-based user rating correction process for interactive recommendation systems, Multim. Tools Appl. 65 (2013) 119–132.
[25] J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: 14th Conference on Uncertainty in Artificial Intelligence (UAI), 1998.
[26] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (2003) 76–80.
[27] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: 10th International Conference on World Wide Web (WWW '01), 2001, pp. 285–295.

[28] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Application of dimensionality reduction in recommender system – a case study, in: WebKDD 2000, 2000.
[29] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (2004) 5–53.
[30] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, J. Mach. Learn. Res. 10 (2009) 2935–2962.
[31] J.F. Huete, J.M. Fernández-Luna, L.M. de Campos, M.A. Rueda-Morales, Using past-prediction accuracy in recommender systems, Inf. Sci. 199 (2012) 78–92.
[32] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowl.-Based Syst. 23 (2010) 520–528.
[33] L. Baltrunas, F. Ricci, Locally adaptive neighborhood selection for collaborative filtering recommendations, in: AH 2008, 2008, pp. 22–31.
[34] I. Gunes, C. Kaleli, A. Bilge, H. Polat, Shilling attacks against recommender systems: a comprehensive survey, Artif. Intell. Rev. (2012).
[35] B. Li, L. Chen, Z. Xingquan, Z. Chengqi, Noisy but non-malicious user detection in social recommender systems, World Wide Web 16 (2013) 677–699.
[36] D. Kluver, et al., How many bits per rating?, in: Proceedings of the Sixth ACM Conference on Recommender Systems, 2012, pp. 99–106.
[37] D. Cosley, S.K. Lam, L. Albert, J.A. Konstan, J. Riedl, Is seeing believing?: how recommender system interfaces affect users' opinions, in: Proceedings of CHI '03, 2003.
[38] X.H. Pham, J.J. Jung, N.T. Nguyen, Integrating multiple experts for correction process in interactive recommendation systems, J. Univ. Comp. Sci. 19 (2013) 581–599.
[39] R. Hu, P. Pu, Exploring relations between personality and user rating behaviors, in: Proceedings of the 1st Workshop on Emotions and Personality in Personalized Services (EMPIRE at UMAP), 2013.
[40] I. Cantador, I. Fernández-Tobías, A. Bellogin, Relating personality types with user preferences in multiple entertainment domains, in: Proceedings of the 1st Workshop on Emotions and Personality in Personalized Services (EMPIRE at UMAP), 2013.
[41] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study of their impacts, Artif. Intell. Rev. 22 (2004) 177–210.
[42] S. Dooms, T. De Pessemier, L. Martens, MovieTweetings: a movie rating dataset collected from Twitter, in: Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys, 2013.
[43] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, MyMediaLite: a free recommender system library, in: Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011), 2011.
[44] P. A. Chirita, Nejdl, W., Zamfir, C., "Preventing Shilling Attacks in Online Recommender Systems", in 7th annual ACM international workshop on Web information and data management, 2005, pp. 67–74.
[45] R. Burke, B. Mobasher, C. Williams, R. Bhaumik, Classification features for attack detection in collaborative recommender systems, in: 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 542–547.
[46] M. Mazurowski, Estimating confidence of individual rating predictions in collaborative filtering recommender systems, Expert Syst. Appl. 40 (2013) 3847–3857.
[47] S. Schelter, C. Boden, V. Markl, Scalable similarity-based neighborhood methods with mapreduce, in: Proceedings of the 6th ACM Conference on Recommender Systems, 2012, pp. 163–170.
[48] W. Zhou, H. Zhang, Correlation range query for effective recommendations, World Wide Web (2013).
[49] S.-T. Park, et al., Naïve filterbots for robust cold-start recommendations, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 699–705.
[50] S. Moon, P.K. Bergey, D. Iacobucci, Dynamic effects among movie ratings, movie revenues, and viewer satisfaction, J. Market. 74 (2010) 108–121.
[51] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Inf. Retrieval 4 (2001) 133–151.
[52] C.N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: 14th International Conference on World Wide Web, 2005, pp. 22–32.