pISSN 1976-913X
eISSN 2092-805X

J Inf Process Syst, Vol.9, No.3, September 2013 http://dx.doi.org/10.3745/JIPS.2013.9.3.435

A Regularity-Based Preprocessing Method for Collaborative Recommender Systems

Raciel Yera Toledo*, Yailé Caballero Mota** and Milton García Borroto***

Abstract—Recommender systems are popular applications that help users to identify items that they could be interested in. A recent research area on recommender systems focuses on detecting several kinds of inconsistencies associated with the user preferences. However, the majority of previous works in this direction just process anomalies that are intentionally introduced by users. In contrast, this paper is centered on finding the way to remove non-malicious anomalies, specifically in collaborative filtering systems. A review of the state-of-the-art in this field shows that no previous work has been carried out for recommendation systems and general data mining scenarios, to exactly perform this preprocessing task. More specifically, in this paper we propose a method that is based on the extraction of knowledge from the dataset in the form of rating regularities (similar to frequent patterns), and their use in order to remove anomalous preferences provided by users. Experiments show that the application of the procedure as a preprocessing step improves the performance of a data-mining task associated with the recommendation and also effectively detects the anomalous preferences.

Keywords—Collaborative Recommender Systems, Inconsistencies, Rating Regularities

1. INTRODUCTION

Recommender systems help users to identify items that they could be interested in. Modern applications usually present to the users a huge collection of items like movies, books, web pages, scientific papers, books, and so on. This makes it difficult to select the appropriate items according to each personal interest. Recommender systems appear as a tool to mitigate the effect of this information overload.

Given a set of users and items, and for each user the preference degrees about a subset of items, Gunawardana and Shani [1] categorize recommender systems into three classes, which are based on the recommendation task that they are designed for. These classes are as follows: 1) those that suggest a list of top preferred items for a specific user, 2) those that suggest a list of items that maximize the profits of the item provider, and 3) those that predict the preference value (generally numeric and also identified as the rating value) for a specific user over an unknown item.

A lot of work have been done to propose solutions for these problems [2, 3]. Considering the kind of data involved, recommendation systems can be grouped in the two main paradigms of

Manuscript received April 2, 2013; first revision May 6, 2013; accepted July 21, 2013.
Corresponding Author: Raciel Yera Toledo
* Knowledge Management Center, University of Ciego de Ávila, Cuba (yeratoledo@gmail.com)
** Department of Computer Science, University of Camagüey, Cuba (yaile.caballero@reduc.edu.cu)
*** Bioplants Center, University of Ciego de Ávila, Cuba (milton.garcia@gmail.com)

content-based and collaborative filtering-based systems. Content-based recommendation systems [4, 5] use features associated with items (i.e., the actors and director of a movie) to characterize the specific preferences for each user and then employ this information to perform the preferences prediction or items recommendation. In contrast, collaborative filtering recommendation systems [6, 7] do not depend on information beyond the rating values to perform the same task. This approach assumes that users who agreed on preferred items in the past will tend to also agree in the future. Following this supposition, the methods in this group predict preferences or recommend items for a given user based on the behavior of other similar users regarding the items that they have chosen.

Most of the methods developed for recommendation scenarios assume each rating as a ground truth value. However, Amatriain, Pujol, and Oliver [8] show that users are inconsistent when they provide ratings in the system, and that this could negatively affect the prediction accuracy. Furthermore, recent reviews [9] have showed that an important challenge in recommendation systems research is related with data quality, because the data underlying the recommendations may be corrupted, or just plain wrong.

Managing data inconsistencies has also been an open problem in more general data mining scenarios, and many data preprocessing methods have been developed to solve this issue. Methods for data preprocessing are organized into the following categories [10]: 1) data cleansing, which is applied to remove noise and correct inconsistencies in the data; 2) data integration, which merges data from multiple sources into a coherent data store; 3) data transformation, which improves the accuracy and efficiency of mining algorithms; and 4) data reduction, which reduces the size of data by eliminating redundant features or by clustering objects.

Data cleansing has been a particularly important research area as it is needed in order to deal with inconsistencies and boosting the effectiveness of different mining tasks. Most of the work on data cleansing is grouped into the following two main categories: those that remove noisy information from the original dataset, and those that just detect and correct it in such a way that the information is not lost. In addition, the majority of approaches have been focused on the supervised learning scenario.

In this direction, Zhu and Wu [11] provided a survey on the impact of noise in various supervised classification methods, showing that two categories of noise, class noise and attribute noise, could appear in this framework. Class noise is associated with labeling errors in the class attribute for a specific object, and it can be associated with contradictory examples and instances labeled with the wrong class. On the other hand, attribute noise is represented by errors that are introduced to attribute values, which makes it more related to the problem associated to this contribution.

Many studies have been performed to deal with class noise [12-14]. These studies have shown that in many cases, the removal of objects that have been erroneously tagged could result in the classification accuracy being improved. However, the attribute noise treatment has been considered a more difficult task to perform, and currently there are few studies centered on this area, as compared to how many have been done on class noise [15-17].

In addition, the majority of advances performed in attribute noise handling indirectly use class information to perform the detection and correction of anomalies [11]. Although some of these proposals could be adapted to scenarios that go beyond supervised classification, this process is not straightforward. Certainly, to the best of our knowledge, there are less bodies of work

centered on noise handling that could be applied to alternative data mining scenarios like the recommendation framework, which has already been presented.

In this paper we are proposing a novel approach to detect and correct inconsistent preference values in collaborative recommendation datasets. We have assumed the hypothesis that there are several implicit groups of users that share the same preference degrees over an important amount of items, and that there are several implicit groups of items that share the same preference degrees over several users. As such, we extracted regularities that characterize groups of users or groups of items. Then, we tagged as the stranger or more anomalous preferences those that were covered by only a few regularities. After that we were able to predict a new value for them by using the preferences associated with similar users regarding the current user. We would assert that the preferences that have been detected as being anomalous, do not represent the criteria associated to the majority of the users and items, and that this preferences could affect the accuracy of any recommendation method associated.

The paper is organized as follows: in Section 2 we present previous work that has been carried out in relation to the treatment of inconsistencies in collaborative recommendation systems and in general data mining. Section 3 presents the regularity-based correction framework. Section 3.1 formally defines what rating regularities means and proposes a method to obtain them. Section 3.2 presents how to use these regularities to detect anomalous ratings in the collaborative filtering dataset. Section 3.3 proposes a strategy to correct them and Section 3.4 presents an analysis on the computational cost of the proposal. Section 4 describes the plans and development for an experimental protocol to verify the effect of the correction approach on the recommendation accuracy in different scenarios, and its ability to detect noisy preferences and to perform its task in a short amount of time. Finally, we present our conclusions in Section 5.

2. RELATED WORK

In this section we present previous work that has been carried out in relation to the treatment of data inconsistencies in general data mining frameworks and specifically for recommendation scenarios. In the case of general data mining, we will center on attribute noise handling, because it is more connected to the current problem of inconsistencies treatment in recommendation systems, as compared to the issue of class noise handling, which we have already mentioned.

Data quality and noisy data handling in data mining have been a topic of interest for many years. Zhang [18] reviewed two kinds of noisy handling methods presented in the literature in this area. These two methods are as follows: 1) those that remove noisy data, and 2) those that detect and correct noisy data. The second one is a better strategy because the resulting dataset could preserve an important part of the original information, but this information is more associated with the ideal noise-free case.

By taking a supervised classification scenario as a starting point, Teng [15] proposed a model to handle the possible erroneous feature values or class labels. He did so working under the assumption that there is some relationship patterns among the feature attributes and the class attribute. To identify and correct noise from an attribute (e.g., A_i) the mechanism switches A_i with the class and uses all other attributes plus the class to train a classifier, which is used to predict the "correct" value of A_i. Zhu et al. [16] extended this idea and further proposed a method to rank the potentially noisy instances by their impact in the classification process. The impact-

sensitive instance ranking takes the information-gain ratio as the evaluation criterion to calculate the impact of each suspect instance on the learning system. Zhang et al. [19] have presented a novel framework that unifies error detection, error correction, and data cleansing to build an aggressive classifier ensemble for effective learning from noisy data, where the classifier ensemble is built from the data that has been preprocessed by the data cleansing and correction methods. Zhang and Wu [20] have also proposed the inference of associated corruption rules to simulate a common noise formation process in real world data and state that they can be used for systematic noise inference and elimination, which enhances the data quality for supervised learning. Also, Marcus et al. [21] have presented a method that is based on ordinal association rules for detecting objects that contain potential errors in the data. Finally, Val Hulse et al. [17] introduced an approach for detecting instances with attribute noise and demonstrated its usefulness by using two different real-world software measurement datasets. Considering all of these bodies of work, only the last three methods do not depend on class labels to perform the corrections. However, in the case of [20], the proposal uses both the original and the noisy dataset to learn the rules. In the case of [17, 21], these methods are focused on the detection of noisy instances, and not on the specific piece of data that represents a portion of the instance that has caused the inconsistency. For these reasons, these proposals are not applicable to the recommendation scenarios.

With respect to these global approaches, Zhu and Wu [11] suggested that instead of adopting some "blind" noise handling mechanism, researchers can design their own noise handling approaches to improve data quality according to their own perspective. Following these principles, there have been several, much more domain-oriented research carried out on this topic in the last few years, and the field of recommender systems is one of these domains that has been studied.

The majority of research has focused on the data preprocessing task around recommender systems, which have been centered on handling malicious data that was intentionally inserted by users to bias the performance of the system [22, 23]. However, several bodies of work, like [8], have shown that users could also introduce inconsistencies without negative intentions, and that not a lot of research that is focused on this last scenario has been conducted.

In order to correct these kinds of inconsistencies in the recommender systems' dataset, Pham and Jung [24] presented a model that explores data beyond ratings. It uses item attributes to learn a user preference model, and marks a rating as being "incorrect" if belongs to the current model, but is under a predefined threshold value and under the mean rating for the corresponding user. This approach depends on additional information that is not usually available, and this is an significant drawback.

Li et al. [25] also proposed an approach to detect the top "noisy but non-malicious" users in collaborative filtering recommendation systems. They did so working under the assumption that the ratings provided by the same user on closely related items should have similar values. The removal of these users from the dataset slightly improves the recommendation accuracy. However, this work is centered on the treatment of anomalous users and not on anomalous preferences, which is the goal of our proposed method.

The current work presents a method that employs a pattern-based approach to correct these non-malicious inconsistencies that are introduced by the users.

3. THE REGULARITY-BASED CORRECTION FRAMEWORK

In this section we present the regularity-based framework, which is used to correct inconsistent preferences in collaborative recommendation systems. The method contains the following three main steps: 1) the discovery of rating regularities in users and items (Section 3.1); 2) the detection of anomalous preferences using these regularities (Section 3.2); and 3) the correction of the anomalous preferences detected in the previous step (Section 3.3). We have also included an analysis of the computational cost for this particular method in Section 3.4.

The approach presented in this work is similar to [20] in the sense that it first discovers explicit knowledge and uses this to eliminate data inconsistencies. However, in [20] the authors depended on the original and the noisy dataset to describe (in the form of rules) the noise formation process and they also used these rules to perform the correction. On the other hand, we have discovered regularities that characterize users and items, tagged the data that contradicted them as being noise, and then corrected the noisy values by using a well-known strategy that is associated with the recommendation systems' field.

Our approach also differs from others that deal with noise in the recommendation field (explained above), in the sense that it does not depend on additional information to perform corrections (contrary to [24]), and works at the rating level (contrary to [25]).

3.1 Ratings Regularities: Formalization and Detection

We have used a pattern-based approach to describe the formalization and detection of regularities. A pattern is an expression that is defined in a language, which describes a collection of objects [26]. A pattern is usually expressed as combinations of feature values, like *(Director= Spielberg, Genre = adventure, Year = 2000)* or as logical properties, like *[Director= Spielberg] ∧[Genre= adventure] ∧[Year< 2005]*. The pattern P covers the object x, or the object x supports the pattern P, if the object fulfills the property expressed by the pattern. An important characteristic of the pattern P is the amount of objects from collection X that supports P, which is called the pattern support and it is denoted by *support(P, X)*.

Following this, we have defined a *rating regularity for users* in the recommendation scenario as a pattern that is expressed by the combination of rating values over items. On the other hand, we have represented a *rating regularity for items* as a pattern expressed by the combination of rating values provided by the users.

Formally, a rating regularity for users was a conjunction of terms with the form <*item op value*>, where *item* represents any item in the system, *op* represents a relational operator in the set {=, <, >}, and *value* is a possible value for the preferences of some users for the mentioned item.

In addition, a rating regularity for items could be defined as a conjunction of terms <*user op value*>, where *user* represents any user in the system, *op* represents a relational operator in the set {=, <, >} and *value* is a possible value for the preferences associated with the mentioned user, over some items in the system.

To present practical examples for these definitions, we used the examples show in Table 1, which shows a classical collaborative filtering scenario with several users (in rows), items (in columns), and some rating values for the corresponding users and items (intersections of rows and columns).

Table 2 presents rating regularities for users that have been extracted from this scenario. In the case of the users' regularities, the first one (Item4 = 2 and Item5 > 3) has the support = 2,

because it is covered by users, User2 and User6. The second one (Item1 = 5) has the support = 3, because it is covered by User1, User3, and User 6. On the other hand, for the regularities of items, the regularity User4< 4 and User6 = 5 receives the support = 2 (verified in Item3 and Item5). The last one (User1 > 3 and User6 = 5) is verified in Item1, Item3, and Item5, and it has the support=3.

The exhaustive set of regularities for a dataset that is similar to Table 1 could easily be found using an Apriori-like method [27]. In order to get just the relevant regularities, we must obtain only those regularities with high support values. Then, we have to define a minimum support threshold for this task. Figure 1 presents an overview of a method to find the regularities. It must be executed twice. The first time is so that regularities for items can be obtained, and the second one is for users.

Initially, this approach generates all of the possible regularities of length 1 (containing just one simple term) that verify the support constraint and stores them in a specific set. After this generation step, we have proposed that an initial filter, which checks if there is more than a simple term for the same item (or user) and the same operator, be applied to avoid redundancy. In this case we leave just the term that has the major support and remove the others. In Table 1, Item1 has two possible values (2 and 5), which implies that the possible simple terms associated to it are {(Item1≤2), (Item1≥2), (Item1=2), (Item1≤5), (Item1≥5), and (Item1=5)}. By checking this set, the initial filter will determine that (Item1≤2) and (Item1≤5) contain the same item and the same operator. It will then remove one of them and leave the term that has the main support. We also applied this procedure for (Item1≥2) and (Item1≥5).

After that, we took the new set without any redundancies as the starting point. We used an Apriori approach to perform several iterations to obtain regularities larger than 1. In this procedure, for each step k, we took the regularities generated in the previous one (with length k-1) as input and used them as a base to generate a new set of candidate regularities with length k. We then stored the ones that satisfied the same support constraint definitive regularities. Also, like Apriori, our process stops when there are no regularities that satisfy this constraint at a specific step.

Table 1. Typical scenario for collaborative filtering

	Item1	Item2	Item3	Item4	Item5
User1	5		4	5	5
User2		3		2	4
User3	5				
User4			3		2
User5	2	5			
User6	5		5	2	5

Table 2. Regularities extracted for the scenario in Table 1

For users	Item4 = 2 and Item5>3	support = 2
	Item1 = 5	support = 3
For items	User4 < 4 and User6 = 5	support = 2
	User1 > 3 and User6 = 5	support = 3

Regularity_Generation(minSup, users, items)
1. For each item (or user), each possible operator, and each possible value, generate all possible terms with length 1 that verify the support constraint.
2. Check if more than one simple term with the same item (or user) and operator exists, and in these cases, just remain the one with the major support.
3. Taking the set of simple terms as a starting point, obtain regularities larger than 1 using an Apriori-like approach.
4. Filter the regularities obtained, verifying that for each regularity, there is no other one that is a subset of it.

Fig.1. Overview of the method to generate regularities in the recommendation scenario

This scenario also implies redundancy because it could contain pairs of regularities like (Item1 = 5) and (Item1 = 5 and Item2 ≤ 3), where the terms that contain one of them represent a subset of the terms that correspond to the other one. For this reason, this final set must be filtered again. Then, we assume that larger regularities are better than smaller ones for expressing the behavior of a group of objects. As such, we are proposing a simple strategy to reduce the original set into another one that verifies that for each regularity there is not another one representing a subset of the first one. We initially proposed that this task being performed through an iterative process, where for each regularity it checks the entire set and removes the regularities that represent a subset of the current one. In the example presented, this approach retains the regularity (Item1 = 5 and Item2 ≤ 3) and removes (Item1 = 5).

Structures like the ones presented in Table 2 have been used to represent knowledge on different domains and for different purposes. It is worth noting that there are some recent works on the use of similar structures to build accurate and understandable supervised classifiers [28, 29]. Particularly in the field of recommendation systems, there are works that use rules to represent the knowledge behind the dataset [30]. Rules are closely related with the concept of regularity, which is presented in this work.

3.2 Detecting Anomalous Ratings through Regularities

The regularities extracted for users and items represent common behaviors that are associated to a group of users or items in the recommendation system. If a specific rating value implies that a certain user or item is covered by a specific regularity, it means that there is a group of users or items that shares the same criteria associated with this preference. As such, we can then conclude that this preference is not anomalous because several users (or items) share it. On the other hand, if a specific regularity does not cover a certain user or item, then it means that the preference or preferences (ratings) that are specifically causing this situation must be analyzed as possible anomalies.

Following these criteria, we have presented an approach to find anomalous ratings through regularities. Our approach (Figure 2) first detects regularities for users using the method presented in Figure 1. It then initializes an error matrix to store the degree of the anomaly for the rating (if the rating exists) associated with each user and item (lines 02-03). After that, we performed two similar procedures for the regularities for users and for items. In the first case, we explored all of the users for every user regularity (lines 04-05). If the corresponding user *u* did not satisfy the current regularity (line 06), we then explored its single terms to determine which preference values in *u* implied that it does not verify the regularity (lines 07-08). We marked these preferences as possible anomalies and incremented their associated values at the corre-

```
FindAnomalousRatingsPerUser(n, minSup, users, items)
01 regForUsersandItems = Regularity_Generation(minSup, users, items)
02 for each user u and item i
03 errorMatrix[u, i]=0;
04 for each regularity ru for users in regForUsersandItems
05          for each user u in users
06                    if ru is not verified for user u and there are preference values for u in all items in ru
07                              for each single term st in ru
08                                        if st is not verified for user u
09                                                  errorMatrix[u, st.item]+=ru.support;
10 for each regularity ri for items in regForUsersandItems
11          for each item i in items
12                    if ri is not verified for item i and there are preference values for i in all users in ri
13                              for each single term st in ri
14                                        if st is not verified for item i
15                                                  errorMatrix[st.user, i]+=ri.support;
16 anomalousItemsPerUserList={}
17 for each user u
18           itemsWithAnomalousRatingForU = {top n items i with larger values in errorMatrix[u, i], for user u}
19           anomalousItemsPerUserList[u.id]=itemsWithAnomalousRatingForU
20 return anomalousItemsPerUserList;
```

Fig. 2. Method for detecting the top n anomalous ratings per user

sponding user and item in the error matrix by adding the support that is associated to the corresponding regularity (line 09). This same procedure is performed in regularity for items. In this case, we explored all of the items for every item regularity (lines 10-11). If corresponding item *i* did not satisfy the current regularity (line 12) we then explored its single terms to find which preference values in *i* imply that it does not verify the regularity (lines 13-14). We then marked these preferences as possible anomalies and added the support of the current regularity to the value associated with the current user and item in the error matrix (line 15). Finally, we employed a user-based strategy to obtain the top anomalous preference values. For each user, we selected and retrieved the n values associated with the items with larger values in the error matrix (lines 16-20) as anomalous.

Taking the scenario presented in Table 1 as an example and considering just the regularities presented in Table 2, the detection process works as follows: evaluating the first regularity for all users (Item4 = 2 and Item5 > 3), the regularity is not verified for User1, and there are explicit ratings for this user and for the corresponding items in the current regularity (Item4 and Item5). Then, there are possible anomalies in this case and we have to check each single term in the regularity (lines 07-09 in Figure 2). It the first term (Item4 = 2) is not verified for this user (in this case User1 set the rating=5 for Item4) we must consider this situation as evidence for considering this rating to be anomalous. We then increment the value in errorMatrix[1,4] (for User1 and Item4). For User2 the regularity verifies the current regularity, so we do not have to do anything. User3 does not satisfy this regularity, but there are also no preference values for User3 in all of the items associated with the regularity (condition in the line 06 of the figure 2), and then there is no action to repeat. This last situation repeats for User4 and User5, and finally User6 verifies the current regularity, like in the case of User2.

When repeating this work for the second regularity (Item1 = 5) we had to analyze just User5 for where its only term does not satisfy the regularity, and then we had to increment the value in errorMatrix[5, 1].

The analysis of the third regularity (User4 < 4 and User6 = 5), which is the first regularity for items, is performed as follows: in the case of Item1 there is no rating value for User4 and Item1, and so there is no action to do. This situation repeats for Item2, where there are no values for User4 or for User6. It also repeats for Item4, where there are no values that are specifically for User4. In the case of Item3 and Item5, both are covered by the regularity, so there is no action to do.

To check the fourth regularity (User1 > 3 and User6 = 5), Item1, Item3, and Item5 are covered by the regularity. In the case of Item2, it does not contain preferences for both users and so there is no action to do. Finally, in the case of Item4, it does not satisfy the regularity because the rating of User6 for Item4 has the value = 2, and not the value = 5. For this reason, we incremented the value in errorMatrix[6, 4] by adding the support of the regularity.

In summary, the detection process using the scenario presented in Table 1 and the regularities presented in Table 2 returns as anomalous preferences that are associated with the following users and items pairs: 1) User1 and Item4 with the anomaly degree = 2; 2) User5 and Item1 with the anomaly degree = 3; and 3) User6 and Item4 with the anomaly degree = 3.

3.3 Ratings Correction

In the previous section we presented a method to detect anomalous preferences per user in collaborative filtering environments. Collaborative filtering predicts rating values for a given user based on past preferences and on the criteria of peers. As such, we are proposing using this same point of view to deal with the anomalous preferences detected with the previous step and for predicting for each case a new value for the corresponding user and item. If the new value was different enough as compared with the older one (detected as anomalous) we performed a replacement by setting the new value for the current user and item.

Figure 3 presents this correction strategy. The approach initially invokes the method shown in Figure 2, and then iterates for each user the list of items whose ratings were detected as being anomalous (line 02-03). Then, for the current user and item, it predicts a new rating value using a traditional collaborative filtering method (line 04). It replaces the old value with the new one if the difference between them exceeds a threshold diff (lines 05-06).

To predict new ratings (line 04), we selected a popular memory-based approach that is based on users (identified as Pearson's user-based collaborative filtering [31]). It employs Pearson's correlation coefficient to find similar users (Equation 1) and an aggregation function that combines their preferences (Equation 3) to obtain the unknown rating. We used this approach because it performs well in the prediction task, and does not require the additional generation of knowledge to calculate the unknown ratings. In the equations (1), (2), and (3), $r_{u,i}$ represents the available rating for the corresponding user u and item i, and $P_{u,i}$ represents the new prediction.

```
Correction(diff)
01 anomalousItemsPerUserList=FindAnomalousRatingsPerUser(n, minSup, users, items)
02 for each user u
03          for each item i in anomalousItemsPerUserList[u.id]
04                    newValue= predictRating(u,i)
05                    if(ratings[u,i]-newValue> diff)
06                              ratings[u,i]= newValue;
```

Fig. 3. Method for correcting anomalous ratings

This method depends on an internal parameter, which is the amount of nearest users that are to be considered for calculating the new rating, which we have named, *k-internal*. We will refer to it in our experimental section.

$$w(u,v) = \frac{\sum_{i \in I}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r_u})^2 \sum_{i \in I}(r_{v,i} - \overline{r_v})^2}} \quad (1)$$

$$\overline{r_u} = \frac{1}{|I_i|}\sum_{i \in I_i} r_{u,i} \quad (2)$$

$$P_{u,i} = \overline{r_u} + \frac{\sum_{v \in Neighbors}((r_{v,i} - \overline{r_v})w(u,v))}{\sum_{v \in Neighbors}|w(u,v)|} \quad (3)$$

3.4 On the Computational Costs Associated with our Method

Our proposal is based on an Apriori approach, and for this reason it tends to have a high computational cost. In this section we will provide a brief analysis on the computational complexity of our method.

In the case of the method to generate regularities (presented in Figure 1), the first step linearly depends on the product of the amount of users and items, which also represents the possible amount of ratings (O(|U|*|I|), where |U| represents the amount of users and |I| the amount of items). Step 2 can be easily implemented inside of Step 1, without an increment of additional computational complexity. On the other hand, Step 3 focuses on finding regularities for users and items and it demands two executions of an Apriori-like approach that exponentially depends on the amount of users and items [32]. This step could undoubtedly affect the efficient execution of the presented procedure. However, real recommendation scenarios are always very sparse, meaning that although they usually have many users and items, there are also relatively few ratings. It could imply that there are few regularities with high support, and that their length tends to be short. These facts could allow for an efficient Apriori implementation to find them quickly, despite the exponential complexity of this method. In the experimental part of this work, we empirically verify this assumption by evaluating the execution time of our method by detecting the regularities for users and items. Finally, in analyzing the fourth step of this method in Figure 1, at most it is quadratic in respect to the amount of regularities detected in the previous step (O(|Ru|^2) + O(|Ri|^2), where |Ru| and |Ri| represent the amount of regularities detected for users and items).

As a second stage, the method that finds anomalous ratings per user (Figure 2) first initializes the error matrix (with the cost O(|U|*|I|)), and then performs an iteration for all of the regularities for the users, and an iteration for all of the regularities for the items (lines 02-15). In addition, each regularity is checked term by term for each user and item. Assuming that l represents the length of a regularity, the computational complexity of this segment is O(|Ru|*|U|*l) + O(|Ri|*|I|*l), but taking into account that this length tends to be short, we can reduce this expression to O(|Ru|*|U|) + O(|Ri|*|I|). Finally, for each user, the method retains the top n anomalous ratings (lines 17-20). As such, for each case we have to previously sort the corresponding items to retrieve the top ones. Then, the cost of this segment is O(|U|*|I|*log |I|), considering that n

objects could be sorted with a cost of O(n*log n). In summary, the cost of this stage is O(|Ru|*|U|) + O(|Ri|*|I|) + O(|U|*|I|*log |I|). Like the previous stage, in the experimental section we present an analysis of its execution time.

Finally, the correction method (presented in Figure 3) performs an iteration for all of the users. For each one, it iterates for the top n anomalous ratings and conditionally corrects each value depending on the new one that has been predicted. In this case, the major cost is associated with the traditional collaborative filtering method, which we used to generate the predictions. However, considering that the anomalous ratings represent a small portion of the data, we show in the experimental section how this task could be done quickly.

Table 3 summarizes the computational costs presented in this section.

Table 3. Computational costs associated with each stage

Regularities detection	First and second step	O(|U|*|I|)
	Third step	$O(2^{\|U\|}) + O(2^{\|I\|})$
	Fourth step	$O(\|Ru\|^2) + O(\|Ri\|^2)$
Anomalous ratings detection	O(|Ru|*|U|) + O(|Ri|*|I|) + O(|U|*|I|*log |I|)	
Ratings correction	O(|U|)	

|U|- amount of users, |I|- amount of items, |Ru|- amount of regularities for users, |Ri|- amount of regularities for items

4. EXPERIMENTS

In order to obtain the effects of our proposal, we developed a case study for Movielens, which is a popular dataset in the collaborative recommender systems field. The original version of Movielens is a well-known dataset containing 100,000 movie ratings on 943 users and 1,682 items where each rating is discrete and is in the range [1; 5].

As such, we implemented our method using an efficient Apriori version proposed by Borgelt [33], which is based on a prefix tree representation of the needed counters and uses a double recursive scheme to count the transactions.

We performed our experiments by orienting them in the following different directions: 1) to determine if the application of our method improves the accuracy of the prediction task in the original data, and then contributes to the removal of inherent inconsistencies (Section 4.1). 2) To determine if our method improves the accuracy of the prediction task for data that has been manually corrupted with a specific strategy (Section 4.2). 3) To determine if our method could effectively identify the noisy preferences that are manually introduced in Section 4.2 (Section 4.3). In addition, we focused on determining the average execution time of our method (Section 4.4), and we give the evaluation of its accuracy and execution time in dealing with a larger recommender systems dataset (Section 4.5).

4.1 Evaluating the Effect of our Method For the Original Dataset

To determine if our method corrects anomalous preferences in the original dataset, we proposed that the data be transformed by using our procedure. After that, we evaluated the performance of a traditional data-mining task associated with this scenario and we performed it with and without the transformation. In this case we select the task related with the prediction of un-

known ratings.

We used the experimental protocol that is also described in [1] to prepare the data for this experiment. In this case, the authors have proposed selecting a set of users from the original dataset and to randomly hide n_a items for each user a, where n_a is also randomly selected for each a. These hidden items composed the test set, and the remaining ones were chosen to be the training set.

To determine the accuracy of the prediction method, the work in [1] proposes each test rating to make a new prediction for the corresponding user and item using the algorithm to evaluate the training data. Then, the final accuracy can be calculated using the mean absolute error (MAE) (Equation 4) for all of the predictions made.

$$MAE(f) = \frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} |f(u,i) - r_{ui}| \tag{4}$$

In order to measure the effect of our correction process, we performed the correction on the training set, and afterwards we compared the accuracy obtained by a traditional algorithm that predicts test set ratings by using the transformed data (our proposal) and using the original one (baseline). Finally, to verify if the difference between them is statistically representative, we also used the procedure proposed in [1], which is as follows: 1) for each user in the test set we calculated two MAE values that represented the average of their predictions, which were made by using our proposal and baseline respectively, and we created two sets where each one contains n predictions, respectively, from the n users; 2) we applied a Wilcoxon signed test between these two distributions to determine if the difference is statistically significant.

We evaluated the impact of our correction process over two well-known recommendation approaches: a user-based method that employs neighborhood preferences to make predictions (memory-based) [2], and a dimensionality reduction method that uses matrix factorization techniques (model-based) [34]. For each case, the experimental results are mainly divided into a sensitivity analysis of parameters and performance results. In assessing the quality of the recommendations, we first determined the sensitivity and the best values for the parameters in both scenarios before running the main experiment.

4.1.1 Impact on a User-Based Neighborhood Method

User-based neighborhood collaborative filtering methods were pioneers in the recommendation systems research that was done in the mid-90's [35]. In this section, we will evaluate our approach on a popular recommendation method summarized in [2] and known as UserKNNPearson. It employs a Pearson's correlation coefficient as the similarity function and an adjusted weighted sum of a neighbor's ratings to calculate the predictions.

UserKNNPearson also uses an internal parameter, which is the amount k of the nearest neighbors that needs to be considered. We employed the value k=60, which has been suggested as the default value by some experimental frameworks in the collaborative filtering field, like MyMediaLite [36].

Our approach depends on the following four parameters to perform the correction process: the amount of top anomalous ratings per user to correct (n), the difference threshold between the old and the new rating (diff), the minimum support to obtain the regularities (minSup), and the number of neighbors used by the memory-based method to predict a new rating value when an

anomalous preference is detected (k-internal).

To set an appropriate value for minSup, which obtains a balance between prediction accuracy and performance, we ran several trials and finally set the value as minSup=70. On the other hand, considering that the value of 60 has been assumed to be good as the amount of nearest users in neighborhood methods, we choose k-internal=60. With respect to this last parameter, we kept this value for the rest of the experiments, because we believe that it does not have a direct effect on the behavior of our method, considering that it is just a parameter of the prediction method that is internally employed. We also verified this fact through additional trials.

To obtain the best values for n and diff, we evaluated the behavior of the proposal by modifying the value of n in the range n = [5; 70] in Step 5, and the value of diff in the range diff= [1; 2,5] in Step 0.5. Our results for these trials (Fig. 4) show that the best accuracy was obtained for diff=1.5, and for n=60.

We used these parameter values to compare our approach against the baseline. The results, which are presented in Table 4, show that when using our strategy, the use of corrected ratings improves the behavior of the recommendation method as compared against a baseline that employs original ratings. This baseline is also represented in Figure 4. In order to determine if this result is statistically significant, we applied a Wilcoxon signed test using the protocol described above, given that the difference between both results is significant with p<0,001.

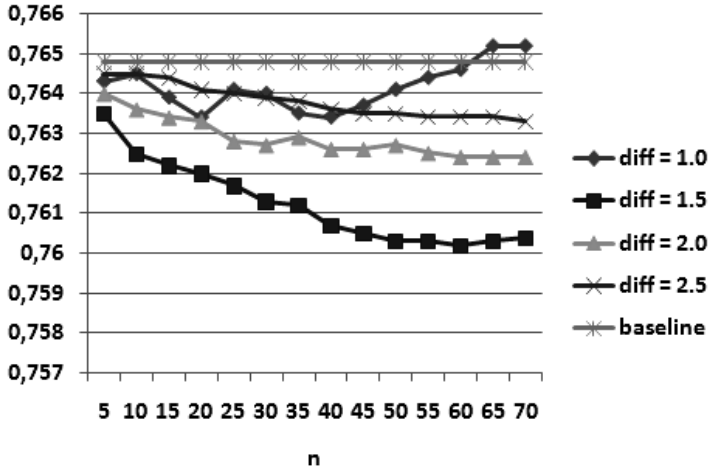

Fig. 4. The behavior of our method with UserKNNPearson for different n and diff values

Table 4. Accuracy associated with UserKNNPearson before and after the rating correction

UserKNNPearson	UserKNNPearson with anomalous rating correction
0,7648	**0,7602**
p<0,001	

4.1.2 Impact on a Matrix Factorization Method

The use of dimensionality reduction methods based on matrix factorization marked a turning point in recommendation systems research at the end of the last decade, because they notably improve the recommendation accuracy [34]. These methods appear as a solution for classical problems like scalability and sparsity, and also indirectly focus on removing small perturbances from the data and on decreasing the impact of anomalous ratings [37]. In this section we describe how we measured the effect of our approach on this scenario, which already contained an implicit manipulation of this kind of noise.

To perform our trial, we adopted the matrix factorization method proposed in [34]. It focuses on gradient descendent techniques to create a lower-dimension space for users and items, and makes predictions using this new space. We used the following default parameter values that were proposed by MyMediaLite [36]: regularization λ_1=0.015, bias regularization λ_2=0, learning rate α=0.01, init_mean=0, and init_stdev=0.1. On the other hand, we performed previous executions to empirically determine the best values in the Movielens scenario for the number of factors in the factorization model (num_factors) and the number of iterations required to build the model (num_iters). We obtained the best performance for num_factors=5 and num_iters=20, and then we used these values in the further experiments.

Using the same strategy presented in Section 4.1.1, minSup=70 and k-internal=60, which are also the same values that we previously used, we found the best values for the parameters n and diff (Fig. 5). Considering that this approach is non deterministic, we ran several trials for each parameter configuration and then averaged them together to calculate the value to compute. Fig.

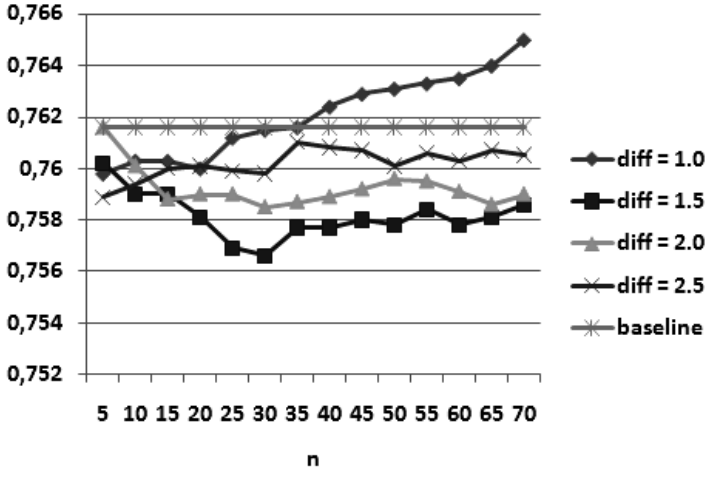

Fig. 5. The behavior of our method with the matrix factorization approach, for different n and diff values

Table 5. Accuracy associated with the matrix factorization approach before and after the rating correction

Matrix factorization approach	Matrix factorization approach with anomalous rating correction
0,7609	**0,7566**
p<0,001	

5 shows the sensitive analysis for both parameters, which obtained the best results for n= 30 and diff=1.5.

We compared our approach with these parameter values against the baseline. Table 5 presents both MAE values that were obtained by our averaging the trial results. Surprisingly, our approach improves the baseline again. This demonstrates that although the matrix factorization methods implicitly remove some anomalies, there are still many preferences whose correction implies a global improvement in the predictions. The results presented show that our approach successfully creates this correction task, which improves prediction accuracy. In addition, they were statistically verified and are significant with $p<0,001$.

4.2 Evaluating the Effect of our Method On the Manually Corrupted Dataset

In the previous section we showed that our approach performs well in an original collaborative filtering scenario. Its use as a preprocessing step for the data, consistently decreases the mean absolute error of two representative approaches performing the prediction task, and this improvement was statistically verified. In this section we pretend to measure the effect of our method on well-predefined noisy data. With this purpose in mind, we manually corrupted the original dataset and compared the performance of the two traditional recommendation methods that we already used in the previous section. We did so by using the corrupted data and by using the data that was transformed through our method.

In this experiment we also used the protocol associated with Section 4.1, with the same partition for the training set and the test set. We used the mean absolute error (MAE) as the evaluation metric. We used a specific strategy to corrupt the training set, and compared the accuracy of the recommendation approaches predicting the test set by using the corrupted training set, and by using the corrupted training set transformed through our approach.

We defined a strategy to manually introduce inconsistencies in the training set. This strategy verifies that the final value for each corrupted rating was significantly different as compared to its original value, and it also verifies that the noise distribution was uniform among all users. In this direction, for each user we selected a set of associated ratings, and changed it depending on its value. Figure 6 presents this corruption strategy, and it receives as parameters the set of users and ratings, and the amount N of ratings to corrupt per user. We used this procedure to alter the training data, setting N in the range [5; 20] with step 5. After we did this, we obtained four train-

```
Corrupt_Dataset(users, ratings, N)
01 for each user u in users
02        Randomly choose N items with ratings associated to the user u
03          for each item i selected
04                    if(ratings[u,i]==1)
05                              ratings[u,i]=5;
06                    if(ratings[u,i]==2)
07                              ratings[u,i]=4;
08                    if(ratings[u,i]==3)
09                              Randomly assign ratings[u,i]=1 or ratings[u,i]=5;
10                    if(ratings[u,i]==4)
11                              ratings[u,i]=2;
12                    if(ratings[u,i]==5)
13                              ratings[u,i]=1;
```

Fig. 6. Dataset corruption process

ing sets with different noise degrees.

In the following sections we will present our results by initially analyzing the sensitive of the parameters n and diff and then finding their best values for each scenario. We then compared the accuracy that we obtained against the baseline.

4.2.1 Impact on a User-Based Neighborhood Method with the Corrupted Dataset

In this section we explain how we used the same prediction method and parameters employed in Section 4.1.1 (UserKNNPearson with k=60), and also set minSup=70 and k-internal=60. Like previous sections, we adjusted the values of n and diff, modifying n in the range [5; 70] with step 5, and diff in the range [1; 2,5] with step 0.5. In this case, we evaluated our proposal over the four datasets that were obtained with the different values of N.

Figure 7 shows the results for these trials. Like our previous experiments, the best results were obtained for larger values of n in all datasets. However, for the noise degrees N=5 and N=10, the MAE value begins to be constant for shorter values of n, as compared against the evaluations associated with the data with N=15 and N=20. This behavior was expected, and

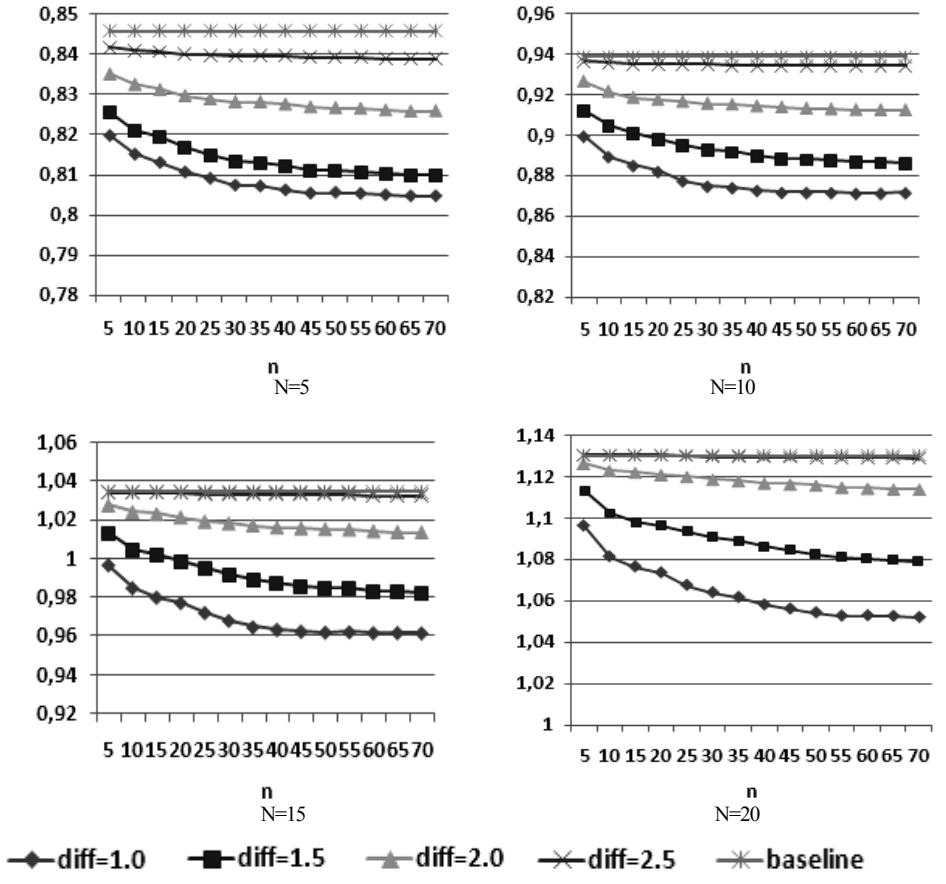

Fig. 7. The behavior of our method with UserKNNPearson and noise for different n, N, and diff values

Table 6. Accuracy associated with UserKNNPearson before and after the rating correction, considering the introduction of noise

	UserKNNPearson	UserKNNPearson with anomalous rating correction
N=5	0,8458	**0,8048**
N=10	0,9389	**0,8712**
N=15	1,0348	**0,9616**
N=20	1,1303	**1,0523**
p<0,001		

allowed us to conclude that the method could effectively detect the ratings that were altered. This is because the data with lower noise need an inferior value of n to obtain the best performance. In contrast to the previous scenario, in this case the global best performances are achieved for diff=1.0 and diff=2.5 with larger noise degrees and method almost does not improve the baseline. Finally, as was expected, the improvement performed by our method at this manually corrupted scenario is larger compared against the improvement associated with the original data.

We selected the parameter configurations associated with the best performance values for each noise degree, and compared them against the baseline. Table 6 shows the comparison results, which show that our strategy indicates an improvement in the prediction accuracy as compared with the related to the noisy and unmodified data. These results were statistically verified using the same protocol presented before, and they were statistically significant with p<0,001.

4.2.2 Impact on a Matrix Factorization Method with the Corrupted Dataset

We also evaluated the performance of the matrix factorization method on the corrupted data and on the corrected corrupted data by using our proposed method. We employed the same parameters used in Section 4.1.2, except for the number of factors and number of iterations. In this case, we previously performed several trials to empirically obtain the best average values for these parameters in the corrupted dataset, and finally set num_factors=5 and num_iter=3. Like previous sections, we also selected minSup=70 and k-internal=60 and evaluated the performance of our method by modifying n in the range [5; 70] with step 5, and diff in the range [1; 2,5] with step 0.5. Just as we did in Section 4.1.2, we ran several trials for each parameter configuration and finally averaged them together to calculate the value to compute. We performed these tasks using the data already employed in Section 4.2.1 that was corrupted. We did so by using the procedure presented in Figure 6, with N in the range [5; 20] with step 5.

The results for these trials, which are presented in Figure 8, are very similar to those obtained by the neighborhood approach. In this case, they represent clearer evidence (compared with the previous section) that a lower noise degree in the data implies that the best results are obtained for a lower value of n. Specifically, for this experiment, for N=5, the results begin to be constant at n=20; for N=10 at n=30; for N=15 at n=40; and for N=20 at n=50. Like previous sections, the best results were obtained for diff=1.0, and we compared them against the baseline for each noise degree (Table 7). For this experiment, we can conclude that the application of our method on the data reduces the prediction error.

In summarizing Sections 4.1 and 4.2, we can conclude that the behavior of our method is

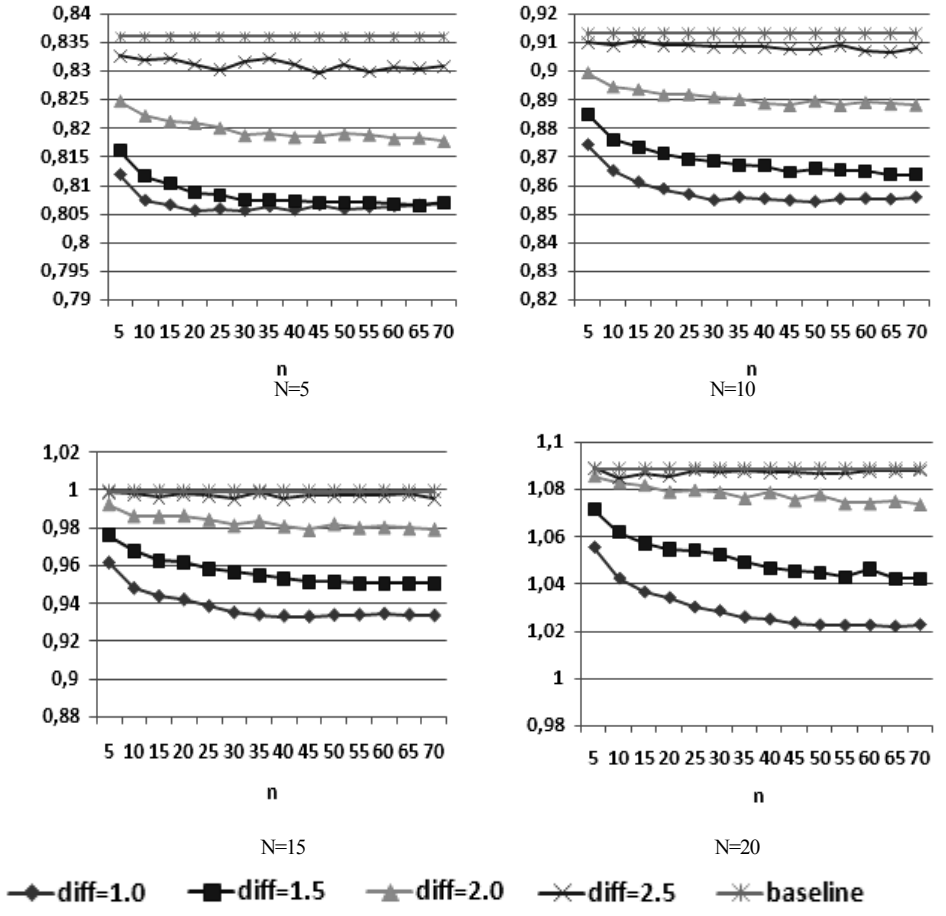

Fig. 8. The behavior of our method with the matrix factorization approach and noise, for different n, N, and diff values

Table 7. Accuracy associated with the matrix factorization approach before and after the rating correction, considering the introduction of noise.

	Matrix factorization approach	Matrix factorization approach with anomalous rating correction
N=5	0,8360	**0,8056**
N=10	0,9131	**0,8544**
N=15	0,9996	**0,9330**
N=20	1,0888	**1,0224**
p<0,001		

different, when we compare the scenario associated with the original data, and the scenario associated to the manually corrupted one. In the first case (Section 4.1), we started with the assumption that the datasets of recommender systems are inherently noisy, and that our proposal contributes to remove this noise by improving the prediction task associated with the corre-

sponding data. The experiments that we performed on a well-known dataset indicate that our procedure suggests a moderate but statistically significant prediction improvement over two other popular recommendation methods. To obtain the best performance, our proposal needs to analyze a medium quantity of ratings per user (30 for the matrix factorization approach and 60 for the neighborhood method). It suggests that although there are several anomalous ratings in the original data, there are not "very noisy" ratings, which verifies that just their being corrected suggests a significant improvement in the prediction of unknown preferences.

On the other hand, in Section 4.2 we corrupted this dataset and altered some of the rating values for each user. In this case, as was expected, the difference between the performance of the predictions associated with this data and the associated with this same data after correction, was bigger than the difference obtained in the previous section. Contrary to the previous stage, the best performance was obtained for diff=1.0 and not for diff=1.5. In addition, we observed a slight correlation between the amount of ratings corrupted per user (N), and the optimal top n noisy ratings to correct per user. This is because the data with lower values in N tends to use low values associated with n, and this is also applicable for the higher values in N. It indicates that our proposal actually tends to first retrieve the "noisiest" ratings, which are represented in this scenario by those that have been manually altered.

4.3 Evaluating the Ability of our Method to Identify Noise that was Manually Introduced

The previous sections have showed that the transformation of the data using our procedure always suggests an accuracy improvement for the prediction task, in both the original and the manually corrupted dataset. In this section we pretend to evaluate the ability of our proposal to detect the noisy ratings that we introduced in Section 4.2. We consider this experiment to be very valuable, because it indicates if the prediction improvements obtained in Section 4.2 were due to the correction of the corrupted preferences or to the correction of other items.

$$precision = \frac{\#tp}{\#tp + \#fp} \tag{5}$$

$$recall = \frac{\#tp}{\#tp + \#fn} \tag{6}$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{7}$$

In this case, we evaluated our method using the F1 metric (Equation 7), which effectively combines the precision and recall metrics [1]. We adopted the confusion matrix presented in

Table 8. Confusion matrix for the correction framework

	Corrected	Not corrected
Corrupted	True-Positive (tp)	False-negative (fn)
Not corrupted	False-Positive (fp)	True-negative (tn)

Table 8, while keeping in mind that a preference could be corrupted or not corrupted, and corrected or not corrected by our approach. In this framework, the precision (Equation 5) is interpreted as being the ratio between the amount of ratings that were corrupted and eventually corrected and the total amount of ratings that were corrected. On the other hand, the recall (equation 6) is interpreted as being the ratio between the amount of ratings that were corrupted and corrected and the total amount of ratings that were corrupted.

We performed the evaluation using the corrupted training sets that we obtained in the previous section, where we randomly altered 5, 10, 15, and 20 ratings, respectively, for each user. In F1 terms, we measured the behavior of our method by only correcting the ratings that had previously been inserted as noise. Like previous sections, we set minSup=70 and k-internal=60, and analyzed the effect of different values for n and diff. Specifically, we set n in the range [5; 70] with step 5, and diff in the range [1; 2,5] with step 0.5.

Figure 9 presents the results of this experiment. It indicates that this method performs the best detection of corrupt ratings for diff= 1.0. Like with our previous experiment, we observed a correlation between the optimal value for the top n ratings corrected per user, and the amount N

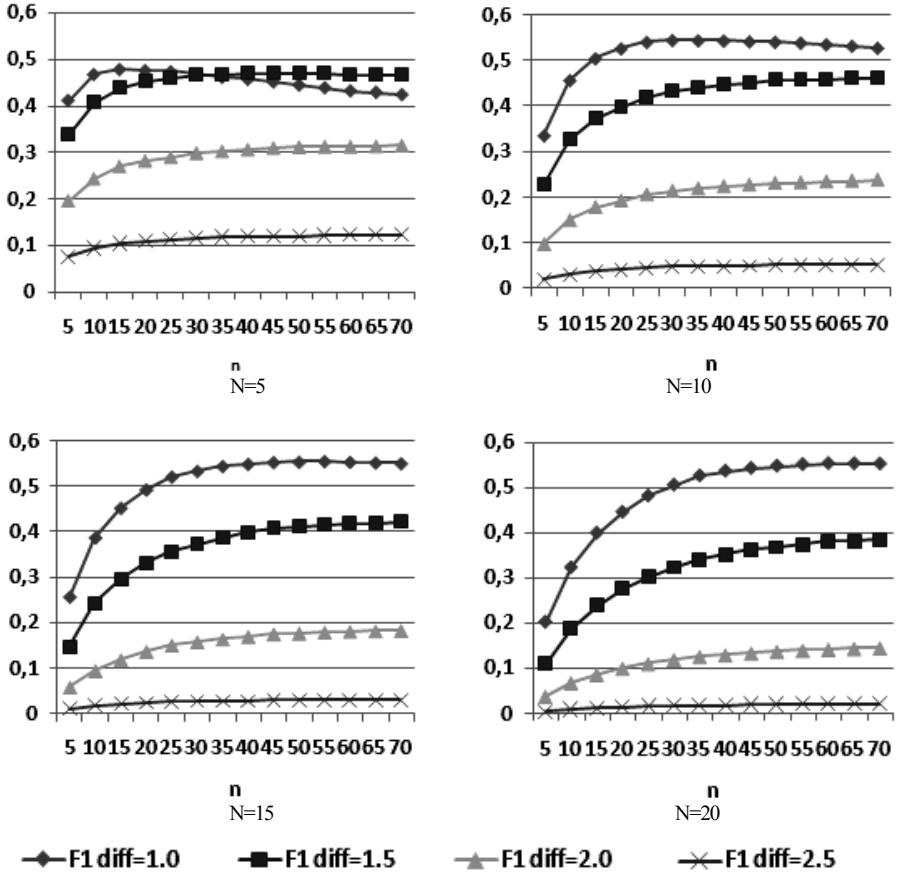

Fig. 9. The accuracy of our method in detecting corrupted ratings, for different n, N, and diff values

of ratings corrupted per user. Considering this performance at diff=1.0, for N=5, and N=10 the best F1 values were obtained for n=15 and n=30, respectively. For bigger values in n, the values of F1 decrease. On the other hand, for N=15 and N=20, the accuracy of the method does not receive a remarkable increment for n>45 and n>55, respectively. This indicates our method already corrects an important amount of altered ratings just for these values. These results are aligned with those described in Section 4.2, where we got the best MAE values for lower values at n, in datasets with lower noise degrees.

4.4 About the Execution Time of the Preprocessing Method

Our method is based on the discovery of regularities represented as frequent patterns, and we have proposed using Apriori-like method to find them. Considering that Apriori tends to have an exponential cost, we previously showed that the execution time of our method could become a significant problem. However, we also mentioned that by taking into account some special features associated to recommender systems datasets (like sparsity), the proposal should run relatively quick.

In this section we will evaluate the execution time of our proposal and will perform this task separately for each method step (regularities detection, anomalous ratings detection, and ratings correction). The execution of the former two steps could be affected by the amount of regularities generated, and for this reason, we will focus on how to determine these steps by modifying the scale of the parameter minSup. The execution time of the third step strongly depends on the amount of ratings that need to be corrected, and so in this case we will evaluate how this step scales by varying the top n ratings that need to be corrected per user.

This experiment was developed by using the original version of MovieLens on a platform containing the Microsoft Windows XP Professional Operating System, an Intel Pentium ® Dual Core 3.06 GHz processor, and a Kingston DDR3 4 Gb RAM memory. For the analysis of the first and second step, we evaluated the parameter minSup in the range [60; 120] with step 10, and left n=60. We selected this range because for minSup >120 the method did not discover any regularities, and for minSup<60 it takes a longer time to finish. Finally, in the third step we evaluated n in the range [10; 50] with step 10, and left minSup=70.

Figure 10 presents the result of this trial. The execution time was measured in seconds. Specifically, Figure 10a (regularities detection) shows that our method combined with the Apriori implementation proposed in [33], could efficiently find regularities for users and items in a relatively short period of time. Figure 10c (rating correction) presents the time employed by the neighborhood method to calculate new ratings, but excludes the time needed to previously calculate the Pearson's correlation coefficient between all users. This data is necessary for obtaining the new ratings, but it is not directly associated to our method because usually it is calculated in real-time systems. For this reason we excluded this execution time from our evaluation.

In summary, the results show that our method could process the current dataset relatively quick. When using our software/hardware platform, the method always finishes in around 6 seconds in the worst case scenario. Taking into account that our proposal is a preprocessing procedure that has been conceived for periodical use on the systems and always as a background stage, we evaluate as very appropriate the time employed by our method.

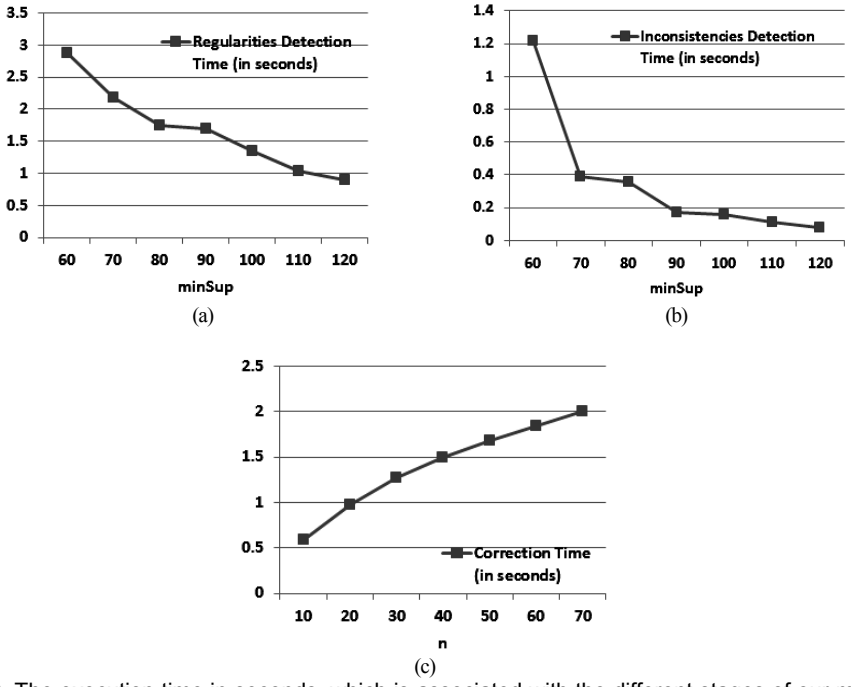

Fig. 10. The execution time in seconds, which is associated with the different stages of our method. a) The detection time for regularities b) The detection time for inconsistencies c) Correction time

4.5 Evaluating the Performance of our Approach in a Larger Dataset

In this work, we have evaluated the performance of our approach using the data in the original Movielens dataset version, which is also known as Movielens 100K. It contains 100,000 ratings provided by 943 users on 1,682 items. The presented results prove that the application of our method as a preprocessing step suggests an improvement in the ratings prediction stage.

We have also previously showed that our method could have a significant computational cost depending on the amount of users and items, but in addition we mentioned that the high sparsity associated with the ratings implies that the proposal runs quickly in practical scenarios. We corroborated this fact in the previous section, where we showed that in the worst-case scenario, our method completes its task in just a few seconds.

In this section, we evaluate the performance of our approach in a larger dataset with the following two criteria: the prediction accuracy after the application of the method (in MAE terms), and the execution time of the method. Specifically, we will use another version of Movielens, known as Movielens 1M, which containing around 1,000,000 ratings that have been provided by more than 6,000 users on approximately 4,000 items.

To evaluate the prediction accuracy, we performed an experimental setup similar to the one we presented in Section 4.1. This setup does not consider the ratings corruption and uses the same procedure that was described in the beginning of Section 4.1 to build the training and test sets. To choose the value for minSup, we selected minSup=430, because it approximately represents the same proportion of users and items as compared with the minSup value that was used

in the experiments with Movielens 100K. On the other hand, we selected k-internal=60 since the previous trials in this scenario concluded that this value is appropriate to use as the amount of nearest neighbors in this dataset. On the other hand, to adjust the best values for the parameters n and diff, we evaluated several executions using different values by considering both recommendation methods (UserKNNPearson and the matrix factorization approach). We obtained the best performance for n=5 and diff=2.0 with the UserKNNPearson method (with k=60), and for n=50 and diff=1.5 with the matrix factorization method (with num_factors=10 and num_iter=30). As was expected, for both cases the performance associated to our preprocessing technique outperforms the baseline, which is represented as the accuracy obtained when using the uncorrected dataset (Table 9).

To evaluate the execution time, we used the same protocol and the same hardware-software platform that we employed in Section 4.4. We varied the minSup in the range [370; 490] with step 20, and n in the range [10; 70] with step 10. In this case, the detection time for regularities and the detection time for inconsistencies also quickly performed their tasks. However, the time

Table 9. Accuracy before and after the rating correction in the dataset, Movielens 1M

	Our method	Our method with anomalous rating correction
UserKNNPearson	0,7532	**0,7494**
Matrix factorization approach	0,7218	**0,7190**
	p<0,001	

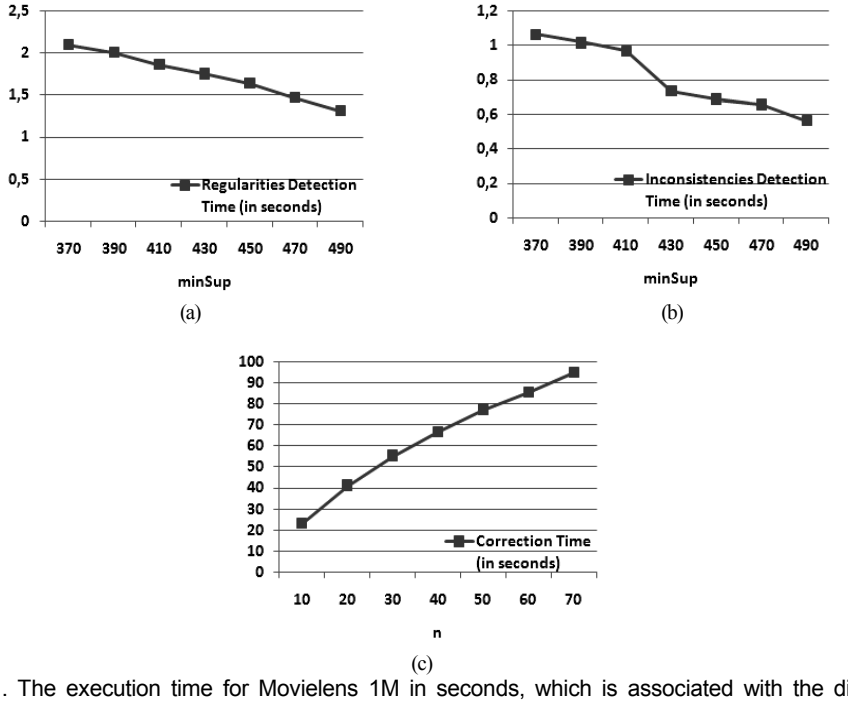

Fig. 11. The execution time for Movielens 1M in seconds, which is associated with the different stages of our method. a) The detection time for regularities b) The detection time for inconsistencies c) Correction time

associated to the correction task notably increased for larger values of n. For these cases, the method takes 100 seconds to correct all of the ratings that have been detected as anomalous. However, by keeping in mind that the method was conceived to be executed as the background in a real system we can classify this time as being an appropriate amount. Like previous sections, Figure 11 presents the execution time of the three stages of our method.

5. CONCLUSION

In this paper we have presented a framework to perform the correction of anomalous ratings in collaborative filtering recommendation systems. Our work adopts the position that users are inconsistent when they rate items, and that this could negatively affect the prediction accuracy. Following this, we proposed an approach to discover rating regularities and considered the ratings that contradicted these regularities as being noisy. We presented a way to correct these noisy values. We also showed that by using our proposal as a preprocessing step, the prediction accuracy of two popular approaches in collaborative filtering is significantly improved in both the original dataset and the manually corrupted one. In addition, our approach effectively discovered the ratings that were manually altered in the corrupted case. Our future work will be focused on proposing alternative methods to obtain more useful sets of regularities that improve accuracy and on defining new forms to represent regularities by possibly using uncertainty approaches.

REFERENCES

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

[2] 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.

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

[4] P. Lops, De Gemmis, M., Semeraro, G, "Content-based recommender systems: state of the art 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] C. Desrosiers, Karypis, G. , "A comprehensive survey of neighborhood-based recommendation methods.," in *Recommender systems handbook* L. R. F. Ricci, B. Shapira, P.B. Kantor Ed., ed: Springer, 2011, pp. 106-144.

[7] W. H. Jeong, Kim, S.J., Park, D.S., Kwak, J., "Performance Improvement of a Movie Recommendation System based on Personal Propensity and Secure Collaborative Filtering," *Journal of Information Processing Systems,* vol. 9, pp. 157-172, 2013.

[8] 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.

[9] J. A. Konstan, Riedl, J., "Recommender systems: from algorithms to user experience," *User Modeling and User-Adapted Interaction,* vol. 22, pp. 101-123, 2012.

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

[11] X. Zhu, Wu, X., "Class Noise vs. Attribute Noise: A Quantitative Study of Their Impacts," *Artificial Intelligence Review,* vol. 22, pp. 177–210, 2004.
[12] C. E. Brodley, Friedl, M.A., "Identifying and Eliminating Mislabeled Training Instances,," presented at the 13th National Conference on Artificial Intelligence (AAAI'96), 1996.
[13] D. Gamberger, Lavrac, N., Dzeroski, S., "Noise Detection and Elimination in Data Preprocessing: experiments in medical domains.," *Applied Artificial Intelligence,* vol. 14, pp. 205-223, 2000.
[14] J. D. Van Hulse, Khoshgoftaar, T.M., "Class noise detection using frequent itemsets," *Intelligent Data Analysis,* vol. 10, pp. 487-507, 2006.
[15] C. M. Teng, "Correcting noisy data," presented at the Proceedings of the Sixteenth International Conference on Machine Learning (ICML' 99), 1999.
[16] X. Zhu, Wu, X., Yang, Y., "Error detection and impact-sensitive instance ranking in noisy datasets," presented at the Ninetheenth National Conference on Artificial Intelligence (AAAI'04), 2004.
[17] J. D. Van Hulse, Khoshgoftaar, T.M., Huang, H., "The pairwise attribute noise detection algorithm," *Knowledge and Information Systems,* vol. 11, pp. 171-190, 2007.
[18] Y. Zhang, "Noise tolerant data mining," Ph.D. Thesis, The Faculty of the Graduate College, University of Vermont, 2008.
[19] Y. Zhang, Zhu, X., Wu, X., Bond, J.P., "Ace: An aggressive classifier ensemble with error detection, correction and cleansing.," presented at the Seventeenth International Conference on Tools with Artificial Intelligence (ICTAI'05), 2005.
[20] Y. Zhang, Wu, X., "Noise modeling with associative corruption rules.," presented at the Seventh IEEE International Conference on Data Mining (ICDM'07), 2007.
[21] A. Marcus, Maletic, J.I., Lin, K.I., "Ordinal Association Rules for Error Identification in Data Sets," presented at the 10th International Conference on Information and Knowledge Management (CIKM'01), 2001.
[22] 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.
[23] I. Gunes, Kaleli, C., Bilge, A., Polat, H., "Shilling attacks against recommender systems: a comprehensive survey," *Artificial Intelligence Review,* 2012.
[24] H. X. Pham, Jung, J.J., "Preference-based user rating correction process for interactive recommendation systems," *Multimedia Tools and Applications,* vol. 65, pp. 119-132, 2013.
[25] B. Li, Chen, L., Xingquan, Z., Chengqi, Z., "Noisy but non-malicious user detection in social recommender systems," *World Wide Web,* 2012.
[26] G. Piatetsky-Shapiro, Frawley, W.J., *Knowledge Discovery in Databases*: AAAI/MIT Press, 1991.
[27] R. Agrawal, Srikant, R., "Fast algorithms for mining association rules," presented at the VLDB, 1994.
[28] M. García-Borroto, Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Medina-Pérez, M.A., Ruiz-Schulcloper, J. , "Lcmine: An efficient algorithm for mining discriminative regularities and its application in supervised classification.," *Pattern Recognition,* vol. 43, pp. 3025-3034, 2010.
[29] M. García-Borroto, Martínez-Trinidad, J.F., Carrasco-Ochoa, "Fuzzy emerging patterns for classifying hard domains," *Knowledge and Information Systems,* vol. 28, pp. 473-489, 2011.
[30] C. W. K. Leung, Chan, S.C.F., Chung, F.L., "A collaborative filtering framework based on fuzzy association rules and multi-level similarity," *Knowledge and Information Systems,* vol. 10, pp. 357-381, 2006.
[31] C. Desrosier, Karypis, G., "A Comprehensive Survey of Neighborhood-based Recommendation Methods," in *Recommender Systems Handbook*, F. R. Ricci, L.;Shapira,B.;Kantor,P., Ed., ed, 2011, pp. 107-145.
[32] C. Borgelt, "Frequent item set mining," *WIREs Data Mining Knowl Discov,* vol. 2, pp. 437-456, 2012.
[33] C. Borgelt, Kruse, R., "Induction of Association Rules: Apriori Implementation," presented at the 14th Conference on Computational Statistics (COMPSTAT), 2002.
[34] Y. Koren, Bell, R. M. Volinsky, C., "Matrix factorization techniques for recommender systems.," *IEEE Computer,* vol. 42, pp. 30-37, 2009.
[35] J. Breese, Heckerman, D., Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering," presented at the 14th Conference on Uncertainty in Artificial Intelligence (UAI), 1998.

[36] C. N. Ziegler, McNee, S. M., Konstan, J. A., Lausen, G. , "Improving recommendation lists through topic diversification.," presented at the 14th International Conference on World Wide Web, 2005.
[37] 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.

Raciel Yera Toledo

He received the BSc. and MSc. degrees (Hons.) in Computer Science from University of Informatics Sciences, Cuba, and University of Ciego de Ávila, Cuba, in 2010 and 2012 respectively. He is currently a PhD. student in Computer Science at Las Villas Central University, Cuba. He is a lecturer of Computer Science at Knowledge Management Center, University of Ciego de Ávila. Its current research interests rely on recommender systems, knowledge discovery from the World Wide Web, and its application in e-learning scenarios.

Yailé Caballero Mota

She received the Ph.D. degree in Computer Science from Las Villas Central University, Cuba, in 2007. She is currently the dean of Faculty of Computer Science in University of Camagüey, Cuba, and the head of the Artificial Intelligence Research Group. She has published widely in leading journals and international conferences proceedings. She is a young member of Cuban Academy of Sciences, and she is the recipient of the Finlay Medal 2012, for her outstanding contributions to the science in Cuba.

Milton García Borroto

He received the Ph.D. degree in Computer Science from National Institute of Astrophysics, Optics and Electronics, Mexico, in 2010. He is currently a researcher in the Computer Science Laboratory, at the Bioplants Center, University of Ciego de Avila, Cuba. He has published widely in international journals, like Pattern Recognition, Knowledge and Information Systems, and AI Review. His research interests are pattern recognition, intelligent systems, machine learning and biometrics.