The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-06267-7

Natural noise management in collaborative recommender systems over time-related information

Francisco J. Baldán[1,2] · **Raciel Yera**[2] · **Luis Martínez**[2]

Accepted: 23 May 2024

Abstract

Recommender systems are currently a suitable alternative for providing easy and appropriate access to information for users in today's digital information-overloaded world. However, an important drawback of these systems is the inconsistent behavior of users in providing item preferences. To address this issue, several natural noise management (NNM) approaches have been proposed, which positively influence recommendation accuracy. However, a major limitation of such previous works is *the disregarding of the time-related information* coupled to the rating data in RSs. Based on this motivation, this paper proposes two novel methods, named SeqNNM and SeqNNM-p for NNM focused on an incremental, time-aware recommender system scenario that has not yet been considered, by performing a classification-based NNM over specific preference sequences, driven by their associated timestamps. Such methods have been evaluated by simulating a real-time scenario and using metrics such as mean absolute error, root-mean-square error, precision, recall, NDCG, number of modified ratings, and running time. The obtained experimental results show that in the used settings, it is possible to achieve better recommendation accuracy with a low intrusion degree. Furthermore, the main innovation associated with the overall contribution is the screening of natural noise management approaches to be used on specific preferences subsets, and not over the whole dataset as discussed by previous authors. These proposed approaches allow the use of natural noise management in large datasets, in which it would be very difficult to correct the entire data.

Keywords Time-aware noise correction · Natural noise management · User inconsistencies · Collaborative filtering · Recommender systems

Extended author information available on the last page of the article

Published online: 08 July 2024

1 Introduction

Recommender systems (RS) currently represent an appropriate solution for the problem associated with efficient information access in today's digital world [4, 32, 36, 84]. Specifically, collaborative filtering (CF) has been highlighted as a very appropriate paradigm to implement these systems because they are able to produce accurate recommendations requiring a minimum amount of information [24].

User-user collaborative filtering, as a pioneer recommendation approach, was initially introduced by Resnick et al. [65] as a way to predict the user's preferences by using the information associated with similar peers associated with a community of people. Later, Sarwar et al. [71] extended the idea behind user similarity and defined item-item CF algorithms that are based on item-based similarities and outperform user-user methods in terms of accuracy and scalability [22]. Koren et al. [40] popularized a new paradigm for CF, proposing dimensionality reduction methods that notably improve the performance of traditional methods. In addition, a lot of notable research has been developed in order to do a deeper exploration of user ratings to enhance the recommendation process [11, 20, 74, 77, 79, 81, 88, 90].

In this way, although the focus on recommender systems research began in the 1990 s [66], it has been evolving across the time in a very active and prolific research field in which currently there are several hot topics such as the item recommendation from implicit feedback [64], the use of deep learning [92], the context-aware recommendation [3], the group recommendation [51], the cross-domain recommendation [18], and the explainability [72], among other popular research efforts.

On the other hand, the preprocessing process of inconsistent user preferences in RS has also become an emerging field of study, mainly focused on movie recommendations [47, 59, 86]. Several authors have stated that user ratings in recommender systems are intrinsically inconsistent because of imperfect and even unintentional user behaviors while they express their preferences, limiting their performance with a *magic barrier* [69].

Extant research has provided different examples of the presence of natural noise in recommender systems:

- Amatriain et al. [8] have suggested that preference values should not be regarded as ground-truth values because the rating gathering is a noisy process.
- Pham and Jung [59] pointed out two probable causes for the presence of natural noise in recommender systems datasets, which are: (1) the fact that user preferences change across time, and (2) the users are imprecise when they provide rating values.
- Said et al. [69] and Kluver et al. [35] have indicated that users' imprecision can be caused by personal conditions, social influences, emotional states, or certain rating scales.
- Yera et al. [86] have presented an illustrative example of natural noise, where a low rating is considered noisy if the corresponding user usually evaluates

positively most of the items, and the associated item has been voted high by the majority of the items.
- Yera et al. [85] also present an illustrative example of natural noise, where the noise degree of a rating is characterized by the number and weights of the identified user's behaviors/regularities that the rating contradicts or does not verify. In this way, it is identified as noisy, a rating that causes the user to not follow a pattern/regularity with high support.

To overcome these issues, some works have been proposed in the last few years, focusing on detecting, removing, or correcting naturally noisy ratings by using their own rating information or information obtained from external sources [13, 47, 56, 59, 86, 94]. In addition, there have been studies centered on detecting whole noisy-but-non-malicious user profiles [44].

Previous research has focused on traditional evaluation setups that use ratings to create training and test sets and employ them to evaluate the accuracy of the corresponding method; in this scenario, these methods imply an improvement in the recommendation performance [27]. However, the data associated with a real-time recommender system do not match these settings. In the real-world scenario, preferences are incrementally entered, and therefore, rating sorting begins to play a relevant role here [5, 16, 73]. Furthermore, the system must simultaneously capture this temporal information and provide user rating predictions. Then, the addition of natural noise management into this new scenario requires a solution to several limitations that are connected with the application of the noise correction approach in RS datasets. Some of these limitations are: (1) to explore whether the application of natural noise management in some segment of recent data instead of the whole dataset would be effective in improving accuracy; (2) to explore the magnitude of the associated improvement degree; (3) to identify how this accuracy could vary across different lengths of the rating sequences; and (4) to evaluate other important criteria in natural noise management, such as the intrusion level and the running time, for performing a trade-off with the accuracy, and therefore suggest conclusions in order to use the new approaches in real scenarios. This paper focuses on supporting these issues by exploring approaches for performing natural noise management in such time-related recommendation scenarios.

Specifically, the main novel contributions of this paper in relation to previous proposals and existing similar approaches are:

- The screening of the natural noise management process, tailored to an incremental, time-aware recommendation scenario.
- The development of a comparison protocol between the time-aware natural noise management and the traditional natural noise management approach without the time dimension.
- An extensive evaluation of the time-aware natural noise management performance, using as new rating predictors in a natural noise management context, up to ten different state-of-the-art recommendation approaches. It includes (1) a clustering-based method [25], (2) a basic neighborhood-based method [65], (3) a neighborhood-based method including average deviation [65], (4) a neigh-

borhood-based method that includes biased-based baselines modeling [65], (5) a method based on negative matrix factorization [48], (6) the Koren's basic SVD approach [40], (7) the Koren's SVD approach with temporal information [38], (8) the slope-one approach [43], (9) a baseline only approach predicting the biased-based baseline estimate for given user and item [38], and (10) a normal predictor based on the distribution of the training set, highlighted by Hug, [31].
- The overall evidence is that a natural noise management approach that incorporates time-related information and time windows is able to reduce the method's intrusiveness, decrease the execution time, as well as lead to a similar or improved accuracy.

The paper is organized as follows. Section 2 presents previous studies related to recommender systems and natural noise management in collaborative recommendations and justifies the selection of a specific approach to be used as the base for the following sections. Section 3 describes new approaches for performing the natural noise management task in an incremental, time-aware movie recommendation scenario. Section 4 plans and develops an experimental framework to evaluate the new NNM framework tailored to the time-related context. Finally, we present a final discussion about the obtained results. Section 5 concludes the paper.

2 Preliminaries

In this section, we present the necessary background for easy following and understanding of our proposal. It includes some antecedents for movie recommender systems, for previous works about the management of natural noise or unintentional inconsistencies that users can introduce in RS, as well as a detailed reference to a pioneer work on natural noise management that will be used as the basis for the current proposal.

2.1 Antecedents on basic recommender systems

The movie recommendation domain boosted initially the development of modern recommender systems in the middle of the 1990 s [4, 65]. Two main recommendation paradigms have been modeled over the last 30 years for developing recommendation tools:

- *Content-based recommendation* The basis of content-based recommendation is the use of movie attributes for composing the user and item profiles, considering the relationship between item attributes and the rating values provided by the users [4]. Then, several scoring approaches are employed to recommend the most appropriate movie profiles to each individual user. Some common movie attributes include genre, director, actors, country, year, or each movie's associated tags [19, 58, 83]. In the last few years, several sophisticated approaches have been developed for building the aforementioned user and item profiles. It

includes the use of advanced machine learning algorithms, as well as the use of semantic tools such as ontologies [46, 74].

- *Collaborative filtering-based recommendation* In the case of collaborative filtering, the working principle relies on crowd preferences to suggest movies for the active user. Specifically, it is based on the discovery, in an explicit or implicit way, of users' rating patterns that are similar to those associated with the current user and uses their associated information for the recommendation generation [55]. Two large families of collaborative filtering approaches can be identified: (1) memory-based [55], focused on directly finding appropriate users' neighborhoods for the current user and using the preferences of such nearest neighbors for recommendation generation, and (2) model-based [40], focused on building intermediate models that comprise the preferences of the user's crowd and can facilitate recommendation generation. Collaborative filtering approaches have been very popular in movie recommendation because they can provide accurate recommendations using only ratings and without any additional information, in contrast to content-based approaches that depend on item attributes for an appropriate performance [60].

Across these traditional approaches, in the last few years, movie recommendations have continuously been a relevant research topic. In this way, Deldjoo et al. [21] model a new concept, titled Movie Genome, as a way of alleviating the new item cold start problem in movie recommendation and therefore improving the recommendation accuracy. Kumar et al. [41] introduce a movie recommender system using sentiment analysis from microblogging data, leveraging, in this way, the content-based recommendation paradigm. Widiyaningtyas et al. [80] explore advanced correlation-based similarities between the user profiles for introducing new algorithms for movie recommendation focused on outperforming previous proposals. In a different direction, Chen et al. [17] exploit users' positive and negative profiles and relies on preferences over movies to compose a novel movie recommendation method.

Overall, while most of the available approaches in movie recommendation focus on improving recommendations through the proposal of more sophisticated recommendation algorithms [26], the current paper follows an alternative research path. Specifically, it will focus on the improvement of recommendation accuracy supported by the management of natural noise [50, 85, 86] associated with user preferences and the use of time-related information.

The next subsection focuses briefly on the incorporation of time-related information into recommender systems.

2.2 Antecedent of time-related information in recommender systems

The value of time-related information in recommender systems was early pointed out by Ding & Li, [23], when they presented several weighting approaches to the

basic memory-based collaborative filtering scenario, being used the time where the user's opinion is provided as input for the weights' calculation.

More recently, Campos et al. [12] present a large-scale survey on the use of time-aware recommender systems, illustrating a taxonomy for classifying the developed works on the use of time as a contextual dimension in this scenario. They cover three main work categories: (1) continuous time-aware heuristic approaches, (2) categorical time-aware heuristic approaches, and (3) time-adaptive models.

[75] have also developed a survey in a similar direction, identifying several groups of research trends related to different challenges in the field, such as the following:

- *Time-aware algorithms focused on modeling time as context.* It includes time-aware factorization and time-aware neighborhood models. In the context-aware framework, time features (day, day of the week, working/nonworking hours) were used in the prefiltering, postfiltering, and modeling stages. The main limitation of this approach is related to the way in which the time variable is considered. It is only taken into account as one more variable of the problem, so traditional models are still applied. This modeling makes it especially complex to notice time series behavior of interest, such as concept drift [62], which can affect attributes such as user preferences, product popularity, or product characteristics, among others.
- *Time-dependent algorithms focused on using time as a sequence.* This includes models that attempt to capture the phenomena related to modeling sequential temporal dynamics in recommender systems. Therefore, it deals with issues such as changes and fluctuations in user preferences and item popularity. Here [75] also considered time-dependent neighborhood models (usually implemented through time decay function or sliding window algorithms) and time-dependent factorization models. One of the potential pitfalls of this approach is the increased complexity of the proposed models compared to traditional models [34]. Such complexity leads to a significant increase in running time that is related to the temporal/sequential nature of the proposed processing.

Eventually, Vinagre et al. [75] also point out the separate modeling of short-term and long-term preferences [67], or the bringing to this context of algorithms formerly focused on processing high-speed data streams [45].

More recently, Rabiu et al. [63] presented an updated survey of temporal (time-related) models in recommender systems built on the same framework of Vinagre et al. [75]. Here, the authors suggest the necessity of incorporating change point detection methods across user preferences to improve the exploitation of the temporal dimension, adding time-related deep learning-based methods in this context, and working toward a tailored evaluation strategy for this scenario.

In the last few years, the research line around time-related recommendations has been linked with the topic of sequential recommendation. Quadrana et al. [61] formalize the input of the sequence-aware recommendation problem as an ordered and often timestamped list of past user actions. Furthermore, recently several authors have also introduced several machine learning-related approaches to this research

framework, such as contrastive learning [15], self-attentive neural architectures [93], or knowledge-graphs [30].

In summary, the brief research literature presented in this section suggests that the use of time-related information has been a research goal of the research community. However, it is also important to point out that most of the developed approaches are centered on proposing algorithms directly focused on improving the recommendation performance. It is relevant that there is a lack of work systematically focused on the use of time-related information in RS tasks such as the data preprocessing [7] or the natural noise management [50], according to recent reviews in this field, already referred [61, 75]. This study aims to fill this gap by proposing a time-related natural noise management framework for a movie recommendation scenario.

The next section provides an overview of natural noise management approaches in RS, which is necessary for the introduction of the methods presented in this paper.

2.3 Related works on natural noise management in RS

The preprocessing of inconsistent user preferences, so-called natural noise, is a relatively new research field in CFRS. In this case, we referred to the inconsistencies unintentionally introduced by users due to factors like the change of taste over time, personal conditions, inconsistent rating strategies, or social influences, which, therefore, cause the appearance of a "magic barrier" that affects performance [69]; and excluded those inconsistencies deliberately inserted by some users to bias the behavior of the system [28, 53]. This second kind of inconsistency, also known as malicious noise, is out of the scope of this paper.

The related literature has identified several examples of the consequences of natural noise on the user experience and the system's performance. Being rating gathering a noisy process [8], it is possible that a user could provide a 5-star noisy rating to some item that does not deserve due to lack of attention implying the subsequent, erroneous recommendation, of items linked to the rated one. Furthermore, the current preference over some items, e.g., movies, could be conditioned by issues such as the release date, the advertising associated with the actors, directors, or the movie itself [59]. Therefore, items preferred by some users in the past could not be preferred at present; conversely, some items disliked in the past could be loved now. Additionally, information from social networks could temporally condition the values of the ratings provided by the user [69]. For example, a 5-star item could be voted with two stars if the user reads negative comments about this item. In a different direction, the variation across diverse rating scales in preference gathering systems, such as [0, 5] or [0, 10], creates confusion in the users and then leads to the introduction of natural noise.

The presence of noisy ratings that contradict the common behaviors or regularities of the users [85] would then imply a negative impact on the recommendation accuracy, taking into account that most of the recommendation approaches are built over the identification of common users' behaviors.

O'Mahony et al. [56] introduced the first study that uses the term "natural noise." Here, the authors focus on identifying whether a rating is noise-free or contains

natural noise. For this purpose, they determine the consistency between the original rating value and a new value predicted through a recommendation algorithm for the same user-item pair. Amatriain et al. [8] also consider that the characterization of natural noise is a key element in the RS research field. Initially, they analyze the response provided by traditional recommendation methods in natural noise conditions using data obtained in three different moments: the second one 24 h after the first one and the third one at least 15 days after the third one. This analysis shows that the error prediction considerably varies for each case. Then, the core of the work proposes a user-dependent procedure to remove these inconsistencies by assuming that there are several available ratings associated with the same user about the same item (one rating and several re-ratings). Pham & Jung [59] have proposed a preference-based approach for rating correction in RS. This proposal focuses on the use of item attributes to represent user preferences and the detection and correction of ratings that do not match the corresponding user preference models. Eventually, Li et al. [44] also presented a method for the inconsistencies handling in CF datasets. In this case, their method works at the user level, detecting noisy but non-malicious users whose preferences can affect the recommendation accuracy. Specifically, the proposal assumes that the ratings provided by the same user on closely correlated items should have similar scores. Then, it captures and accumulates the user's contradictions and uses them to remove the top noisy profiles. This removal implies an improvement in the recommendation's accuracy.

More recently, keeping in mind the same goal, the degree of user coherence in RS datasets has been measured using item attributes (e.g., directors, actors), showing that the recommendation accuracy is improved when the users with a lower coherence are discarded [10]. This work is continued by Yu et al. [91], proposing a correction approach for the preferences associated with such low-coherence users. In parallel, Saia et al. [68] presented an approach that uses semantic information to remove incoherent items from user profiles in recommendation scenarios. On the other hand, Yera et al. [86] and Castro et al. [13] proposed a natural noise management method for collaborative RS based on a correction paradigm. In contrast to previous studies, it does not depend on additional information beyond the rating matrix, like item attributes or user feedback. In this case, the method uses a previous classification approach to characterize user and item behavior and detects anomalous ratings based on this classification. Finally, for these tagged ratings, a correction process is performed by calculating a new rating value for the same user-item pair using the remaining ratings and a traditional CF technique. In particular, corrections are made if the difference between the old and new ratings is higher than the threshold. In this case, the predictor used to calculate such ratings was Resnick's user-based CF method with Pearson's similarity (UserKNNPearson) [65].

Over the last few years, several authors have extended the pioneering work developed by Yera et al. [86], being enriched with further computational intelligence techniques and extending the initial ideas. In this way, Zhu et al. [94] take advantage of the correlations between the entropy of the rating data and the prediction uncertainty in terms of evaluation metrics and develops a new denoising algorithm based on fuzzy clustering. The authors assume that the recommendation accuracy is specific to natural noise and that the entropy of an individual

rating dataset indicates the uncertainty derived from noisy data. Furthermore, the fuzzy C-means algorithm is used for noisy rating verification. Recently, Luo et al. [47] presented a new approach for natural noise management in recommender systems that detects natural noise according to the inconsistency between rating behaviors and users' and items' categories in a similar way to Yera et al. [86]. Furthermore, the authors consider the probability that each user belongs to each subcategory and correct the natural noise with threshold values weighted by probabilities.

In parallel, Wang et al. [76] follow the same scheme and proposes an approach that employs fuzzy theory to handle natural noise in RS by classifying ratings into three fuzzy categories characterized by variable boundaries. Subsequently, fuzzy profiles of users and items are constructed to effectively identify natural noise within the ratings. Upon detecting noisy ratings, the authors employ the Maximum Membership Principle to replace them with rating threshold values. Also, Bag et al. [9] re-classify users and items of a system into three classes, namely strong, average, and weak, to identify and correct noise ratings. Subsequently, this study integrates the Bhattacharya coefficient, a well-performing similarity measure for a sparse dataset, with the proposed reclassification method to predict unrated items from the obtained noise-free sparse dataset and recommend preferred products to consumers. In addition, deep learning-based architectures have also been used for natural noise management in RS. Recently, Park et al. [57] proposed an autoencoder-based recommender system for exploiting the ability of both anomaly detection and CF. The proposed system detects natural noise in the rating data based on reconstruction errors after training. By removing the

Table 1 Comparative analysis of existing approaches

Approach	Avoid loss of information	Does not use additional information	Considers a time-related context	Tailored to a group scenario
[56]	–	✓	–	–
[8]	✓	–	–	–
[59]	✓	–	–	–
[44]	–	✓	–	–
[10]	✓	–	–	–
[86]	✓	✓	–	–
[91]	✓	–	–	–
[68]	✓	–	–	–
[13]	✓	✓	–	✓
[94]	✓	✓	–	–
[9]	✓	✓	–	–
[85]	✓	✓	–	–
[76]	✓	✓	–	–
[47]	✓	✓	–	–
[57]	–	✓	–	–
This work	✓	✓	✓	–

detected natural noise, the collaborative filtering approach can predict the unrated ratings using noise-free data.

Table 1 summarizes the described methods in terms of four main features:

- Avoid loss of information: It refers to the fact of avoiding the removal of user preferences across the natural noise management approach.
- Does not use additional information: It refers to the performance of the noise management without the dependency on information beyond the user preference values. Examples of this additional information could be item attributes or tags.
- Considers a time-related context: It refers to the use of rating timestamps or similar time-related variables in the developed models.
- Tailored to a group scenario: It refers to natural noise management models specifically conceived or evaluated in group recommendation scenarios.

Works like [56] and [44], even though they do not depend on additional information beyond the rating values, remove important information from the dataset. Other research like the developed by Pham & Jung, [59], Amatriain et al. [8], Bellogíín et al. [10], Yu et al. [91], and Saia et al. [68], although focusing on rating correction and therefore does not imply information loss, also depend on additional information beyond the rating matrix and therefore could be difficult to apply in some scenarios. Eventually, Yera et al. [85] introduce a regularity-based correction approach that does not depend on additional information but requires the discovery of intermediate knowledge in terms of association rules, which could be difficult to generalize in some scenarios.

In contrast to the abovementioned works, the previous classification-based approach developed by Yera et al. [86] and Castro et al. [13], also featured recently by Bag et al. [9] and Luo et al. [47], correct ratings, does not remove important information from the dataset, and does not depend on additional information such as item attributes. In this way, while most of the considered approaches are centered on individual recommendation, Castro et al. [13] have introduced natural noise management in group recommender systems.

Therefore, considering the advantages of the previous classification-based approach for natural noise management [86], as well as its increasing popularity according to recent works that have continued this approach [9, 47] (see Table 1), the rest of the paper will take the pioneering previous classification-based approach [86], as the base for the current proposal. As presented in Table 1, the current proposal provides a novel feature in managing a time-related context, which contrasts previous approaches that do not consider it. We leave to future work the tailoring to a group recommendation scenario.

2.4 The classification-based approach for natural noise management in RS

The classification-based approach for natural noise management in RS (Fig. 1) was proposed as a way to perform this task without using additional information beyond the user ratings [86].

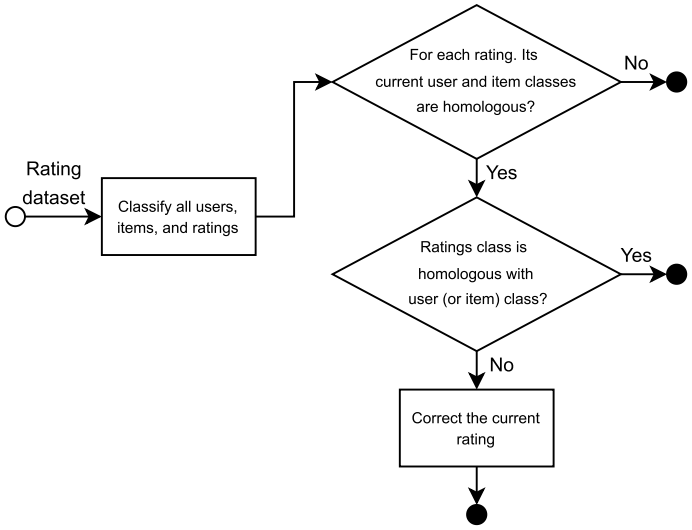

Fig. 1 Global scheme of the previous classification-based approach for natural noise management

This approach comprises two main stages: (1) the detection of possible noisy ratings, and (2) the correction of noisy ratings.

The first stage performs a classification of users and items based on a direct inspection of their ratings, to identify tendencies to have low, medium, or high preferences. Overall, the ratings that do not match those well-identified tendencies are considered as possible noisy. This is the underlying technique behind this stage.

Specifically, each user, item, and rating are, respectively, classified into three possible classes, which are presented in Table 2. Specifically, it focuses on classifying users in the classes benevolent, average, critical, or variable; and items in the classes strongly preferred, averagely preferred, weakly preferred, or variably preferred. Variable and variably preferred classes are used, respectively, for users and items that can not be classified into specific classes. Besides, ratings can be classified as weak, average, or strong, depending on two thresholds. Algorithm 3 (included in Appendix A) shows the pseudocode of this process also included in our new proposals. Moreover, the proposal considers three groups that establish matching among user, item, and rating classes. The proposed method assumes that for a certain rating, if its user and item classes belong to the same group (different from the variable class),

Table 2 Group of 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

Table 3 Classes definition

User classes	Definition
Critical	$\|W_u\| > \|A_u\| + \|S_u\|$
Average	$\|A_u\| > \|W_u\| + \|S_u\|$
Benevolent	$\|S_u\| > \|W_u\| + \|A_u\|$
Item classes	**Definition**
Weakly preferred	$\|W_i\| > \|A_i\| + \|S_i\|$
Averagely preferred	$\|A_i\| > \|W_i\| + \|S_i\|$
Strongly preferred	$\|S_i\| > \|W_i\| + \|A_i\|$

then the rating should belong to the corresponding rating class in the same group. Otherwise, the rating should be classified as a possible inconsistency.

Table 3 presents criteria for classifying users and items using such rating classification. In the case of users, it assumes that for each user u the sets $|W_u|$, $|A_u|$ and $|S_u|$ are the respective sets of weak, average, and strong ratings. Regarding the proportion of ratings in each class, the user is classified as critical, benevolent, or average, and those users who have a similar proportion of the three kinds of ratings are classified as variable users. In the case of items, it follows a very similar approach in relation to users but considers all the ratings associated with the item (see also Table 3). Here, sets $|W_i|$, $|A_i|$, and $|S_i|$ are used as the respective weakly preferred, averagely preferred, and strongly preferred ratings for item i.

The second stage of the proposal is focused on correcting the ratings identified as possible inconsistencies, obtained in the previous stage. Specifically, a new rating value is predicted for each user-item pair associated with the possible noisy rating previously detected. This stage then uses an underlying rating prediction algorithm, which is the well-known Resnick's user-based method with Pearson's similarity (UserKNNPearson) [65], as the former collaborative filtering approach. In each case, if the original rating is sufficiently different from the predicted value, the old rating is replaced with the new one. In the proposal, the difference threshold was set to $\delta = 1$, as this value tends to be the minimum step between two ratings in recommendation scenarios. Algorithm 4 (included in Appendix A) presents this procedure, which is included in our new proposals.

As presented in this section, several authors have pointed out that user preferences evolve over time and that taking this issue into account leads to performance improvement in RS models [15, 30, 75]. It is then necessary to explore how the use of time-related information affects the behavior of this natural noise management model, which has already been justified. Therefore, two new proposals for performing natural noise management in an incremental, time-related recommendation scenario are presented in the next section.

3 Correcting noisy ratings in a time-aware recommendation scenario

The recommendation tasks are intrinsically incremental, taking into account that the ratings stored behind a CF recommender system are provided by users who simultaneously request suggestions from the system itself. However, as presented in the Introduction section, the use of natural noise management approaches to this incremental scenario brings new issues that have not yet been regarded, and to the best of our knowledge, no previous studies have focused on solving this task. Typical natural noise management methods receive as input a set of ratings and optionally additional information about them and return as output the corrected set. Under these circumstances, its deployment in an incremental, time-aware scenario faces troubles like the selection of the ratings set to be corrected across time and the selection of the data that must be considered for the correction process. Taking into account the relevancy of the time dimension and sequential recommendation context as research trends, it is necessary to tailor formerly developed natural noise management models to these new requirements and scenarios. Therefore, the goal of the current study is to screen new models for the natural noise management process contextualized to an incremental, time-related recommendation scenario.

These models use as underlying algorithms the approach for identifying possibly noisy ratings and the approach for correcting noisy ratings. Both algorithms, formerly proposed by Yera et al. [86], have been discussed in Sect. 2.4 and detailed in Algorithms 3 and 4

Furthermore, this work developed a comprehensive experimental procedure over several recommendation approaches, with a higher, more general magnitude

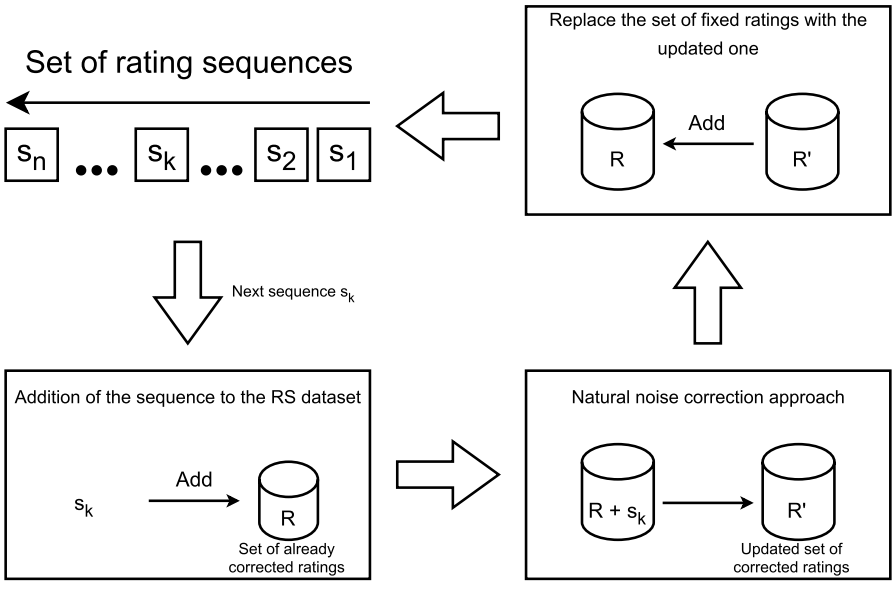

Fig. 2 Overview for the approach for natural noise correction in a sequential scenario

in relation to the previously referred works on natural noise management. Specifically, the current research work will then screen two frameworks for natural noise management in recommender systems, where it is assumed a sequential gathering of the rating data, which is the real context of a deployed recommender system. The next sections describe these approaches.

3.1 Sequential natural noise management in collaborative filtering

In the first stage, we propose a framework, named SeqNNM, that considers the continuous gathering of sequential rating data by RS. Figure 2 illustrates this framework. Herein, it is assumed that a set of rating sequences $s_1, s_2, ..., s_k, ..., s_n$ is continuously gathered by the system. Each newly gathered s_k is first added to the main RS dataset R. Then, the $R + s_k$ dataset is corrected through the mentioned natural noise management approach. From the identification of noisy ratings, following Algorithm 3, and the subsequent prediction of corrected ratings, following the guidelines in Algorithm 4, a processed dataset is finally obtained with the noise corrected based on the available data up to that moment. The sequential processing of data in specific time steps is the main innovation of this proposal. After that, the data reached as output by the NNM approach started to be used as the main data of the recommender system for both the main recommendation generation process and for the subsequent runs of the NNM process. This procedure processes multiple times all the available data, so it is able to correct a large amount of noise through further intrusion into the original data. Algorithm 1 presents an overview of this framework.

Algorithm 1 Pseudocode for the incremental time-aware natural noise management proposal. *seq* method.

Input : R_{data} the sorted set of ratings to be checked for natural noise detection and correction
s_{list} *a list of rating sequences,* $s_1, s_2, s_3, ..., s_k, ..., s_n \in R_{data}$, continuously gathered by the recommender system
$th1$ first classification thresholds
$th2$ second classification thresholds
δ difference threshold
Output: $R_{corrected}$ list of corrected ratings
SequentialNNCorrection(R_{data}, s_{list})

```
R_corrected = { }
foreach new s_k do
    R_temp=R_corrected+s_k
    R_noise=identifyNoiseRatings(R_temp, th1, th2)
    foreach r_ui in R_noise do
        R_corrected=corrrectNoisyRating(r_ui, R_temp, δ)
    end foreach
end foreach
return R_corrected
```

3.2 Sequential natural noise management in collaborative filtering covering the last p rating sequences

The framework for sequential natural noise management presented in the previous subsection has a shortcoming of the high volume of data that is used for natural noise management across the processing of each new sequence, which could affect the time performance of the proposal. To alleviate this drawback, we propose an alternative approach, named SeqNNM-p, in which instead of correcting all data every time that a new rating sequence is processed, it would be corrected only the last p sequences of the most recent ratings in the dataset. This approach significantly limits the data to be processed in each iteration, considerably reducing the final running time and the intrusiveness of the original proposal since the number of instances identified as noise is reduced with a shorter time horizon.

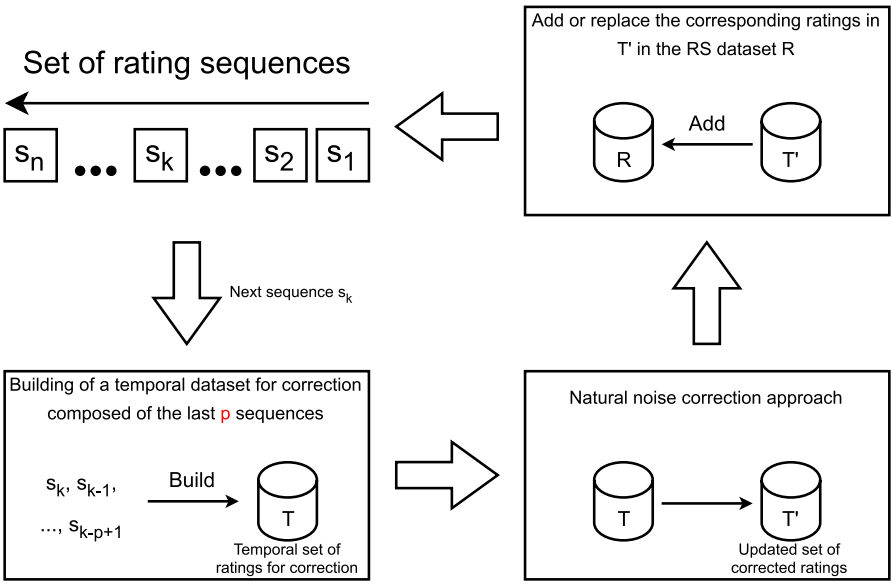

Fig. 3 Overview on an improved approach for natural noise correction in a sequential scenario, considering the correction of the last k sequences

Figure 3 illustrates this approach. Here, once the new rating sequence s_k is gathered, a temporal dataset T containing such a sequence, as well as previous ones. The natural noise management used as the starting point for these models (Sect. 2.4) is applied over this temporal dataset T, and at the last stage, the values of the modified ratings in T are updated in the original dataset R used for recommendation generation. Algorithm 2 screens this approach.

Algorithm 2 Pseudocode for the incremental time-aware natural noise management proposal, considering the correction of the last k sequences. *seqk* method.

Input : R_{data} the sorted set of ratings to be checked for natural noise detection and correction
s_{list} *a list of rating sequences,* $s_1, s_2, s_3, ..., s_k, ..., s_n \in R_{data}$, continuously gathered by the recommender system
$th1$ first classification thresholds
$th2$ second classification thresholds
δ difference threshold
Output: $R_{corrected}$ list of corrected ratings
SequentialKNNCorrection()

```
R_corrected = { }
foreach new s_k do
    R_temp=∪_{i=k-p+1}^{k} s_k
    R_noise=identifyNoiseRatings(R_temp, th1, th2)
    foreach r_ui in R_noise do
        R_temp=corrrectNoisyRating(r_ui, R_temp, δ)
    end foreach
    foreach rating p_ui in R_temp do
        if rating for (u,i) not in R_corrected then
            Add p_ui to R_corrected
        if p_ui ≠ r_ui ∈ R_corrected then
            r_ui = p_ui
    end foreach
end foreach
return R_corrected
```

Overall, the computational cost of both approaches presented in this section depends on two main factors: (1) the cost of the classification-based approach for natural noise management, which is used at the initial step in both approaches, and (2) the cost of the inner approaches for rating prediction. In the first case, considering that full inspection of the rating matrix is necessary, the theoretical cost would be $O(|U| * |I|)$, where U and I are the sets of users and items. However, due to the sparsity of RS datasets, this matrix can be quickly inspected. In the second case, the complexity of the different rating prediction methods varies from methods with

constant time to methods with higher complexity. Moreover, in the experimental section, it will be proved that, in practice, the approach is able to correct several ratings in a short period. In addition, it will be proved how the considered length of the sequence can manage such execution time while maintaining positive values in terms of accuracy for almost all the evaluated settings.

4 Experiments and results

This section executes an evaluation process to measure the impact of the proposed alternatives to natural noise management in an incremental time-aware recommendation scenario. We assume two main criteria for the performance evaluation: recommendation accuracy after the execution of the correction method in the data and the amount of rating modified by the correction process. With this aim, we initially discuss the experimental setup and then present and analyze the obtained experimental findings.

4.1 Evaluation protocol

In this study, we evaluated how our natural noise preprocessing approach increases data quality, thereby affecting recommendation accuracy. Therefore, we will compare the results provided by our sequential approach versus two different cases: the case of not applying natural noise methods and the case of applying the natural noise method identified as baseline [86], but without considering the sequential nature. After applying the selected natural noise method, the recommendation results were evaluated using a fivefold cross-validation approach.

It is important to highlight that the same rating prediction model is used for both the preprocessing natural noise step and the final recommendation. The next subsection further details the prediction models used to evaluate the natural noise management schemes proposed here. For the sequential proposal, it is necessary to simulate a real-world environment. Specifically, the initial training dataset will comprise data for the first ten weeks of the time frame linked to the dataset used in the experiments. The natural noise process is then performed over the entire dataset, adding the next week's information in a sequential manner.

4.2 Models

In order to obtain robust results and to serve as a comparative basis for the state of the art, all prediction models included in the Python Surprise package [31] have been included in the experimentation. The selected models are included in Table 4. It is important to note that each model uses the default configuration set in Surprise.

The approaches proposed in this work, *SeqNNM* and *SeqNNM-p*, are identified for simplicity by *seq* and *seqk*, respectively, in the experimentation carried out.

Table 4 Recommendation algorithms used

Model	Reference
BaselineOnly	[39]
CoClustering	[25]
KNNBaseline	[39]
KNNBasic	A basic collaborative filtering algorithm [65]
KNNWithMeans	A basic collaborative filtering algorithm, taking into account the mean ratings of each user [65]
NMF	[49]
NormalPredictor	Algorithm predicting a random rating based on the distribution of the training set, which is assumed to be normal
SVD	[70]
SVD++	[37, 70]
SlopeOne	[43]

4.3 Datasets and evaluation metrics

Our evaluation protocol selects two different versions of the Movielens dataset [29], which is a popular one in the RS field and additionally has a timestamp for each rating. First, MovieLens100k contains 100,000 movie ratings associated with 943 users, about 1682 items, where each rating belongs to the range [1, 5]. Second, the last 1 million instances of the MovieLens25M dataset, which contains 1,000,000 movie ratings associated with 8715 users, about 5667 items, where each rating is in the range [1, 5]. It is important to highlight that these datasets have been considered state-of-the-art datasets in recommender systems and are currently used by several research works [1, 2, 42, 54].

To evaluate the performance of the proposals, we perform a fivefold cross-validation approach, where 80% of the samples compose the training set and the remaining 20% the test set and measure the recommendation accuracy through widely used metrics such as the mean absolute error (MAE), the root-mean-square error (RMSE), normalized discounted cumulative gain (NDCG) [78], precision, recall, and F1-Score (F1). The NDCG metric [33] relies on discounted cumulative gain (DCG) and it is grounded in the assumption that highly relevant items appearing toward the end of a search result list should be penalized. This is because the graded relevance value diminishes logarithmically in proportion to the position of the result. The formalization of DCG is as follows:

$$DCG_u = \sum_{k=1}^{N} \frac{r_{u,recom_{u,k}}}{\log_2 (k+1)} \tag{1}$$

where $recom_{u,k} \in I$ is the item recommended to user u in k position.

To calculate NDCG, the DCG value needs to be normalized by dividing it by the maximum achievable DCG value, known as $DCG_{perfect}$ [33]. $DCG_{perfect}$

Table 5 Performance measures used for evaluating the recommendation accuracy

Performance measure	Definition
MAE	$\frac{1}{\lvert R_{test}\rvert}\sum_{r_{ui}\in R_{test}}\lvert p_{ui}-r_{ui}\rvert$
RMSE	$\sqrt{\sum_{r_{ui}\in R_{test}}\frac{(p_{ui}-r_{ui})^2}{\lvert R_{test}\rvert}}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-Score	$\frac{2*Precision*Recall}{Precision+Recall}=\frac{2*TP}{2*TP+FP+FN}$

represents an ideal recommendation list where the most preferred items are ranked at the top. The NDCG values for each user are computed as follows:

$$NDCG = \frac{DCG}{DCG_{perfect}} \tag{2}$$

As a final step, the NDCG values associated with individual users are averaged to derive the final reported NDCG value. Additionally, we extract the number of modified values through the natural noise correction process and the running time of the complete experimentation. The definition of the most known performance measures used is included in Table 5.

The intrusiveness of the studied models is another important parameter to be analyzed and is evaluated through the number of values modified by the applied natural noise techniques. In this scenario, a greater number of modified values indicate greater intrusiveness of the method.

Finally, the running time of each proposal is recorded to evaluate its scalability.

4.4 Experimental results

In this section, we present the experimental findings for the specified protocol. To facilitate the reading of this work, we have included a graphical analysis of the most relevant performance metrics of the obtained results. Furthermore, the tables with numerical results present all the metrics included in Sect. 4.3 for the results associated with the models proposed in Sect. 4.2.

4.4.1 MovieLens 100k dataset

The results included in Table 6 show a robust improvement in the recommendation performance when we apply traditional or sequenced natural noise corrections. In the same way, for each recommendation method, the proposed sequential natural noise correction process (*seq*) obtains the best results in most cases. The Baseline-Only method combined with the proposed sequential natural noise process obtained the best results in RMSE, MAE, and precision metrics. The KNNBasic method obtains the best results in Recall and F1-score metrics, but these results are not very

Table 6 Results obtained for MovieLens 100k dataset: no natural noise method (no), baseline natural noise proposal (nn), the sequential proposal (seq), and the sequential method considering the last *k* rating sequences (seqk)

Models	Natural Noise Approach	RMSE	MAE	NDCG	Precision	Recall	F1	Modified Ratings	Running Time (secs)
BaselineOnly	no	0.94397	0.74837	0.91373	0.69851	0.54736	0.61375	0	5.42
BaselineOnly	nn	0.73562	0.56683	0.94282	0.74252	0.5931	0.65943	14527	82.9
BaselineOnly	seq	***0.69145***	***0.52756***	*0.94781*	***0.74274***	*0.60026*	*0.66393*	18801	823.75
BaselineOnly	seqk7	0.82979	0.63833	0.92695	0.71381	0.56736	0.6322	12639	22.04
BaselineOnly	seqk9	0.81738	0.62711	0.92944	0.71457	0.56816	0.63299	13636	26.07
BaselineOnly	seqk11	0.80611	0.61738	0.93161	0.7148	0.56906	0.63363	14538	29.55
CoClustering	no	0.96511	0.75564	0.91146	0.67569	0.52317	0.58971	0	14.39
CoClustering	nn	0.7839	0.59991	0.93614	*0.71232*	0.54387	0.61676	12128	95.89
CoClustering	seq	*0.73695*	*0.55406*	*0.94128*	0.70904	*0.54747*	*0.61784*	17336	930.93
CoClustering	seqk7	0.85769	0.65025	0.92227	0.69189	0.53694	0.60463	12639	36.38
CoClustering	seqk9	0.84457	0.63813	0.92599	0.6917	0.5391	0.60592	13636	40.77
CoClustering	seqk11	0.83104	0.62507	0.92859	0.69702	0.54175	0.60963	14538	44.17
KNNBaseline	no	0.93067	0.73321	0.91707	0.70963	0.55883	0.62525	0	23.46
KNNBaseline	nn	0.78182	0.61532	0.9342	0.72728	0.57847	0.64437	9118	109.05
KNNBaseline	seq	*0.72807*	*0.5677*	*0.94141*	*0.73846*	*0.59383*	*0.65828*	13288	961.72
KNNBaseline	seqk7	0.82601	0.63355	0.92688	0.71363	0.56972	0.63359	12639	41.68
KNNBaseline	seqk9	0.81297	0.62229	0.92989	0.7164	0.57407	0.63736	13636	45.64
KNNBaseline	seqk11	0.80226	0.6132	0.93147	0.71699	0.57367	0.63734	14538	48.93

Table 6 (continued)

Models	Natural Noise Approach	RMSE	MAE	NDCG	Precision	Recall	F1	Modified Ratings	Running Time (secs)
KNNBasic	no	0.97973	0.77363	0.91803	0.71338	0.57295	0.63548	0	19.22
KNNBasic	nn	0.82952	0.64888	0.93443	0.73341	0.59483	0.65686	9727	104.4
KNNBasic	seq	*0.77754*	*0.60621*	*0.94196*	*0.74138*	***0.61219***	***0.67062***	14299	959.35
KNNBasic	seqk7	0.88091	0.67166	0.92788	0.72211	0.60014	0.65549	12639	37.49
KNNBasic	seqk9	0.86868	0.6612	0.93041	0.72	0.60349	0.65659	13636	43.01
KNNBasic	seqk11	0.85936	0.65272	0.9322	0.72193	0.60793	0.66002	14538	45.24
KNNWithMeans	no	0.95108	0.74946	0.91567	0.68056	0.51733	0.58781	0	19.98
KNNWithMeans	nn	0.80108	0.62861	0.93254	0.68745	0.52302	0.59405	9720	106.05
KNNWithMeans	seq	*0.74683*	*0.57921*	*0.93856*	*0.69499*	0.53312	*0.60338*	13959	955.23
KNNWithMeans	seqk7	0.84329	0.64594	0.92451	0.68255	0.53001	0.59667	12639	38.77
KNNWithMeans	seqk9	0.82989	0.63392	0.92739	0.68241	0.53205	0.59789	13636	42.87
KNNWithMeans	seqk11	0.81836	0.62373	0.92885	0.68343	*0.53378*	0.59939	14538	46.03
NMF	no	0.96249	0.75687	0.90787	0.68878	0.5195	0.59224	0	11.77
NMF	nn	0.7939	0.6144	0.92838	0.70282	0.53452	0.60721	11402	101.52
NMF	seq	*0.71873*	*0.54495*	*0.94004*	*0.70567*	*0.54037*	*0.61205*	18591	975.34
NMF	seqk7	0.85485	0.65374	0.91966	0.69491	0.52477	0.59796	12639	33.96
NMF	seqk9	0.84386	0.64306	0.92158	0.69736	0.52753	0.60067	13636	38.16
NMF	seqk11	0.83255	0.63226	0.92417	0.69742	0.52862	0.60138	14538	41.64
NormalPredictor	no	1.51801	1.21871	0.84491	0.56385	0.40898	0.47406	0	4.02
NormalPredictor	nn	1.35557	1.08223	0.86073	0.57832	0.42143	0.48754	15008	74.3

Table 6 (continued)

Models	Natural Noise Approach	RMSE	MAE	NDCG	Precision	Recall	F1	Modified Ratings	Running Time (secs)
Normal-Predictor	seq	*1.14326*	*0.90008*	*0.89593*	*0.6135*	*0.435*	*0.50904*	23706	822.8
Normal-Predictor	seqk7	1.42868	1.14532	0.85411	0.56781	0.4173	0.48106	8841	21.26
Normal-Predictor	seqk9	1.42409	1.14037	0.85299	0.56273	0.40565	0.4714	9607	25.42
Normal-Predictor	seqk11	1.4159	1.13243	0.85334	0.56188	0.40983	0.47394	10119	28.88
SVD	no	0.93613	0.73759	0.91493	0.7011	0.54362	0.61239	0	10.28
SVD	nn	0.81262	0.64443	0.93194	*0.72134*	0.56343	0.63265	7529	78.55
SVD	seq	*0.72243*	*0.56203*	*0.94186*	0.72042	*0.56841*	*0.63543*	15187	826.52
SVD	seqk7	0.83305	0.63869	0.92487	0.70523	0.54711	0.61618	12639	24.79
SVD	seqk9	0.81917	0.6264	0.92756	0.70506	0.5509	0.61849	13636	29.1
SVD	seqk11	0.80833	0.61691	0.92952	0.70614	0.55291	0.62019	14538	32.23
SVD++	no	0.91996	0.72135	0.92007	0.70957	0.55054	0.61999	0	161.33
SVD++	nn	0.76528	0.59833	0.94018	*0.73672*	0.57562	0.64626	9516	278.67
SVD++	seq	*0.69185*	*0.53127*	***0.94812***	0.73353	*0.57863*	*0.64692*	16526	1626.78
SVD++	seqk7	0.8214	0.6287	0.92771	0.70928	0.54968	0.61935	12639	209.55
SVD++	seqk9	0.80695	0.6157	0.93113	0.71169	0.55375	0.62283	13636	225.11
SVD++	seqk11	0.79739	0.60744	0.93202	0.71164	0.55571	0.62406	14538	231.04
Slope-One	no	0.94587	0.74335	0.91307	0.69392	0.5373	0.60563	0	17.9
Slope-One	nn	0.75652	0.58208	0.93877	0.73282	0.56969	0.64102	12353	100.62
Slope-One	seq	*0.72831*	*0.5593*	*0.94202*	*0.73311*	*0.5726*	*0.64297*	14467	945.27
Slope-One	seqk7	0.83611	0.63703	0.92545	0.70573	0.54892	0.61751	12639	36.77
Slope-One	seqk9	0.82315	0.62512	0.92824	0.70779	0.55154	0.61994	13636	40.56
Slope-One	seqk11	0.81254	0.61553	0.93024	0.70708	0.54976	0.61855	14538	44.05

The best cases for each method are highlighted in italic. The best results are stressed in bold

different from those obtained by BaselineOnly. On the other hand, SVD++ obtained the best results for NDCG. This metric is particularly relevant to this type of problem

Table 7 Percentage of improvement (%) of the *seq* proposal vs. the state-of-the-art *nn* proposal for MovieLens 100k dataset

Models	RMSE (%)	MAE (%)	NDCG (%)	Precision (%)	Recall (%)	F1 (%)	Modified Ratings (%)	Running Time (secs) (%)
Baseline-Only	6.00446	6.92800	0.52926	0.02963	1.20722	0.68241	−29.42108	−893.61290
CoClustering	5.98928	7.64281	0.54906	−0.46047	0.66192	0.17511	−42.94195	−870.83685
KNN-Baseline	6.87498	7.73906	0.77178	1.53723	2.65528	2.15870	−45.73371	−781.88837
KNNBasic	6.26627	6.57595	0.80584	1.08670	2.91848	2.09481	−47.00319	−818.95722
KNN-With-Means	6.77211	7.85861	0.64555	1.09681	1.93109	1.57057	−43.61111	−800.76381
NMF	9.46845	11.30371	1.25595	0.40551	1.09444	0.79709	−63.05034	−860.71504
Normal-Predictor	15.66205	16.83099	4.08955	6.08314	3.21999	4.40989	−57.95576	−1007.35500
SVD	11.09867	12.78649	1.06445	−0.12754	0.88387	0.43942	−101.71337	−952.24599
SVD++	9.59518	11.20786	0.84452	−0.43300	0.52291	0.10213	−73.66541	−483.75989
SlopeOne	3.72892	3.91355	0.34620	0.03957	0.51080	0.30420	−17.11325	−839.40943

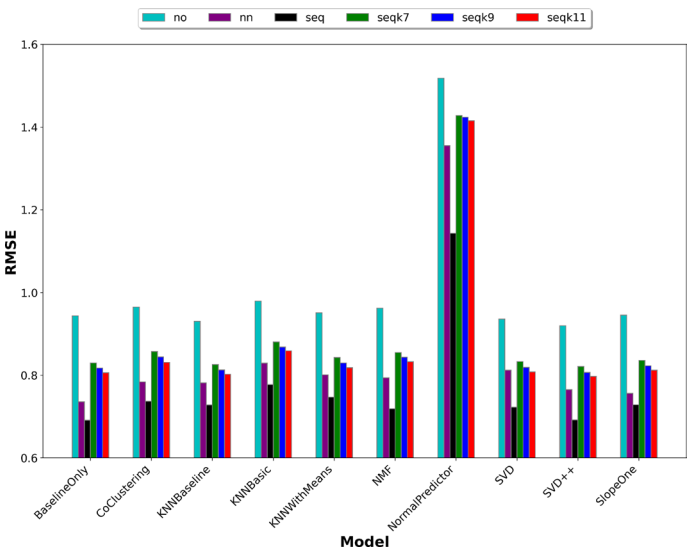

Fig. 4 RMSE results for the multiple models and natural noise approaches selected in the MovieLens 100k dataset

and is gaining importance over time. SVD++ also offers very competitive results in other metrics, especially RMSE, MAE, and precision. BaselineOnly obtains the best results at the cost of being the second most intrusive method after NormalPredictor. It is important to note that methods that are significantly less intrusive than BaselineOnly, such as SVD++ or KNNBasic, are able to provide similar performances.

In order to facilitate the comparison between the *nn* model and the *seq* model, Table 7 shows the percentage improvement obtained in each metric. In this case, a considerable improvement can be seen in all performance metrics in practically all cases. In the case of running time and the number of modified values, we see how *seq* is more time-consuming and intrusive. Both metrics show the cost of obtaining better results.

To analyze the results more clearly, some graphical comparisons have been included.

Figure 4 includes the RMSE results for every model tested and all the natural noise approaches included in this work. The comparison shows how the application of any natural noise correction technique improves the final results obtained using all the methods. Our first proposal, sequential and cumulative natural noise correction (*seq*), provides the best results for all models, with significant improvements over the traditional approach (*nn*). Our second proposal (*seqk*) offers results that are progressively closer to those obtained by the traditional method (*nn*) as the value of *k* increases. This behavior is relevant in massive data or Big Data environments considering that *seqk* works with a reduced subset of data while *nn* needs all available data. Moreover, in the case of SVD, *seqk11* is able to provide better results than *nn*, so in these environments, the *seqk* approach becomes a desirable alternative.

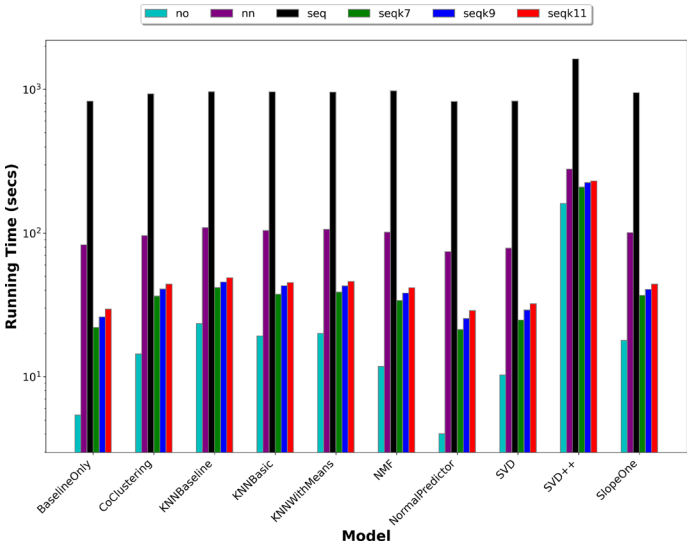

Fig. 5 Running time (secs) results, in logarithmic scale, for the multiple models and natural noise approaches selected in the MovieLens 100k dataset

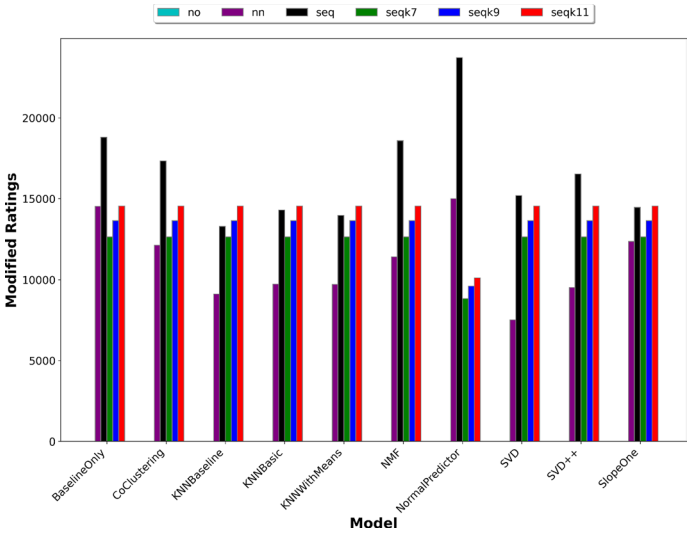

Fig. 6 Number of modified ratings produced for each model and natural noise approach evaluated combination in the MovieLens 100k dataset

The NDCG metric (Table 7) shows a similar behavior to that observed for RMSE. The *seq* approach obtains the best results, *nn* obtains the second place, followed by the different *seqk* approaches, the higher the k, the better the performance. This metric shows reduced differences between *seqk* and *nn*, and it is possible to improve the results of the traditional approach with slight increases in the k parameter. It is important to note the reduction of resources associated with the *seqk* approach, both in time and memory, by processing a subset of the original data at each step.

Figure 5 shows the running time obtained from each approach. The difference between the *seq* approach with respect to the rest is quite clear. This approach requires the longest running time. In the second place, we found the traditional approach (*nn*), while our second proposal (*seqk*) offers a considerable reduction in the running time concerning *nn*. Since the cost in the result performance is reduced, the *seqk* approach provides a robust alternative in environments where time is a constraint to be considered.

Finally, Fig. 6 shows the number of modified values for each model and approach. The *seq* approach is the most intrusive among the models, except for the SlopeOne and KNN-based models. The *seqk* approach is more intrusive as the value of k increases. This behavior shows that a higher data availability leads to higher natural noise detection and, therefore, higher intrusiveness. Re-evaluation of the data when new data are sequentially added also leads to greater intrusiveness, as in the *seq* case.

Table 8 Results obtained for MovieLens 1 M dataset: no natural noise method (no), baseline natural noise proposal (nn), the sequential proposal (seq), and the sequential method considering the last *k* rating sequences (seqk)

Models	Natural Noise Approach	RMSE	MAE	NDCG	Precision	Recall	F1	Modified Ratings	Running Time (secs)
Baseline-Only	no	0.9045	0.71412	0.92298	0.71811	0.56994	0.63551	0	81.92
Baseline-Only	nn	0.7166	0.55467	0.94705	0.75102	0.6053	0.67033	128249	8719.99
Baseline-Only	seq	*0.67968*	*0.52162*	*0.95098*	***0.75287***	*0.61615*	***0.67768***	165506	217044.8
Baseline-Only	seqk7	0.80165	0.6189	0.93435	0.72684	0.58262	0.64679	119606	413.4
Baseline-Only	seqk9	0.79905	0.61663	0.93488	0.72732	0.58339	0.64745	121959	472.42
Baseline-Only	seqk11	0.79576	0.61395	0.93543	0.72697	0.5834	0.64732	124459	558.41
CoClustering	no	0.91263	0.71281	0.92464	0.7037	0.55873	0.62289	0	154.11
CoClustering	nn	0.74229	0.57018	0.94482	*0.73056*	0.58374	0.64895	112804	7666.49
CoClustering	seq	*0.70363*	*0.53261*	*0.94903*	0.72924	*0.58704*	*0.65046*	159584	216431.68
CoClustering	seqk7	0.81639	0.62191	0.93382	0.71153	0.56614	0.63056	119606	512.84
CoClustering	seqk9	0.81084	0.61768	0.93454	0.71241	0.56971	0.63312	121959	564.73
CoClustering	seqk11	0.80871	0.61482	0.93519	0.71262	0.56884	0.63266	124459	698.63
KNN-Baseline	no	0.89624	0.70529	0.92375	0.71945	0.57261	0.63769	0	616.48
KNN-Baseline	nn	0.80037	0.63473	0.93192	0.72558	0.58042	0.64493	55729	8346.76
KNN-Baseline	seq	*0.75403*	*0.59515*	*0.9372*	*0.72954*	*0.58999*	*0.65239*	89148	220434.11
KNN-Baseline	seqk7	0.79971	0.61733	0.9335	0.72539	0.58096	0.64518	119606	936.94
KNN-Baseline	seqk9	0.79711	0.61523	0.9339	0.72647	0.582	0.64626	121959	987.28
KNN-Baseline	seqk11	0.79384	0.6126	0.93454	0.72578	0.58225	0.64614	124459	1067.88
KNNBasic	no	0.93529	0.73738	0.92439	0.71818	0.5988	0.65308	0	747.07

Table 8 (continued)

Models	Natural Noise Approach	RMSE	MAE	NDCG	Precision	Recall	F1	Modified Ratings	Running Time (secs)
KNNBasic	nn	0.83961	0.66558	0.93129	0.72443	0.60933	0.66191	57210	9396.34
KNNBasic	seq	*0.79088*	*0.62373*	*0.93677*	0.72915	***0.61907***	*0.66962*	97594	223299.58
KNNBasic	seqk7	0.84285	0.64746	0.93391	0.72832	0.61471	0.6667	119606	1078.19
KNNBasic	seqk9	0.84034	0.6453	0.93443	*0.72934*	0.61612	0.66797	121959	1167.65
KNNBasic	seqk11	0.8372	0.64272	0.93504	0.72881	0.61685	0.66817	124459	1285.16
KNNWithMeans	no	0.92434	0.73021	0.92135	0.68389	0.53334	0.59931	0	722.12
KNNWithMeans	nn	0.82265	0.65337	0.9305	0.68713	0.53587	0.60214	64782	8877.83
KNNWithMeans	seq	*0.77313*	*0.60956*	*0.93567*	0.68885	0.54257	0.60702	100863	220632.42
KNNWithMeans	seqk7	0.82257	0.63621	0.93127	0.69373	0.54538	0.61067	119606	899.73
KNNWithMeans	seqk9	0.81985	0.63386	0.93176	0.69437	0.54612	0.61138	121959	955.05
KNNWithMeans	seqk11	0.81645	0.63102	0.93243	*0.6951*	*0.54673*	*0.61205*	124459	1037.78
NMF	no	0.91588	0.7201	0.9193	0.69677	0.53433	0.60483	0	131.88
NMF	nn	0.74417	0.57726	0.94004	*0.71794*	0.55391	0.62535	114743	7632.58
NMF	seq	*0.68188*	*0.5179*	*0.94891*	0.71766	*0.55887*	*0.62839*	184003	215615.8
NMF	seqk7	0.81572	0.62808	0.92993	0.70578	0.53999	0.61185	119606	481.98
NMF	seqk9	0.81289	0.62539	0.93041	0.70684	0.54209	0.61359	121959	536.5
NMF	seqk11	0.80971	0.62283	0.93099	0.70592	0.54181	0.61307	124459	613.05
NormalPredictor	no	1.51028	1.21401	0.85059	0.55968	0.40268	0.46837	0	67.79
NormalPredictor	nn	1.35526	1.08332	0.86377	0.56656	0.40661	0.47344	141563	8376.64
NormalPredictor	seq	*1.15084*	*0.90733*	*0.89786*	*0.60137*	*0.42608*	*0.49877*	225176	219088.62
NormalPredictor	seqk7	1.42381	1.14078	0.85739	0.5567	0.40052	0.46587	83481	443.44

Table 8 (continued)

Models	Natural Noise Approach	RMSE	MAE	NDCG	Precision	Recall	F1	Modified Ratings	Running Time (secs)
Normal-Predictor	seqk9	1.42394	1.1409	0.85719	0.55744	0.40201	0.46713	84887	516.02
Normal-Predictor	seqk11	1.42287	1.13982	0.85737	0.55785	0.40191	0.46721	87066	639.33
SVD	no	0.88207	0.69149	0.92781	0.72318	0.56249	0.6328	0	111.93
SVD	nn	0.76815	0.60688	0.94119	0.73858	0.57867	0.64892	67717	7982.44
SVD	seq	*0.67975*	*0.52599*	*0.95112*	*0.73873*	*0.58529*	*0.65312*	152278	215875.73
SVD	seqk7	0.79717	0.6131	0.93402	0.72525	0.56777	0.63692	119606	443.4
SVD	seqk9	0.79468	0.61109	0.93439	0.72634	0.56894	0.63808	121959	520.05
SVD	seqk11	0.79082	0.60806	0.93486	0.7249	0.56773	0.63676	124459	568.96
SVD++	no	0.86704	0.67483	0.93271	0.73233	0.57421	0.6437	0	3048.93
SVD++	nn	0.72654	0.5658	0.94897	*0.7514*	0.59207	0.66228	87500	11406.44
SVD++	seq	***0.66065***	***0.50447***	***0.95588***	0.74983	*0.59762*	*0.66513*	157357	250978.33
SVD++	seqk7	0.78568	0.60084	0.93794	0.73112	0.57334	0.64269	119606	3904.18
SVD++	seqk9	0.78296	0.59857	0.93867	0.73246	0.57374	0.64345	121959	3958.81
SVD++	seqk11	0.78011	0.5961	0.93889	0.73168	0.57429	0.6435	124459	4038.17
SlopeOne	no	0.90424	0.70993	0.92328	0.7088	0.55153	0.62035	0	285.17
SlopeOne	nn	0.72927	0.56432	0.94552	0.73887	0.57676	0.64783	114407	7821.6
SlopeOne	seq	*0.70597*	*0.5449*	*0.94793*	*0.73975*	*0.58208*	*0.65151*	134183	217920.66
SlopeOne	seqk7	0.80414	0.61754	0.93414	0.71629	0.55896	0.62792	119606	631.97
SlopeOne	seqk9	0.80164	0.61529	0.93465	0.71662	0.55938	0.62831	121959	681.4
SlopeOne	seqk11	0.79827	0.61249	0.93518	0.71641	0.55967	0.62841	124459	760.12

The best cases for each method are highlighted in italic. The best results are stressed in bold

4.4.2 MovieLens last 1 M of 25 M rating dataset

In this section, experimentation close to a real use case in a data-intensive environment is performed, allowing us to evaluate the performance and scalability of the proposals.

The results shown in Table 8 show a clear dominance of the SVD++ model with the proposed *seq* approach, obtaining the best results in terms of RMSE, MAE, and NDCG metrics. BaselineOnly, also with the *seq* approach, obtains the best precision and F1-Score results. Finally, the KNNBasic model, with the *seq* approach, obtains the best results in the Recall metric. Although the *seq* approach is more data-intrusive than traditional approaches, the differences in performance results are very significant, as can be seen in Fig. 7.

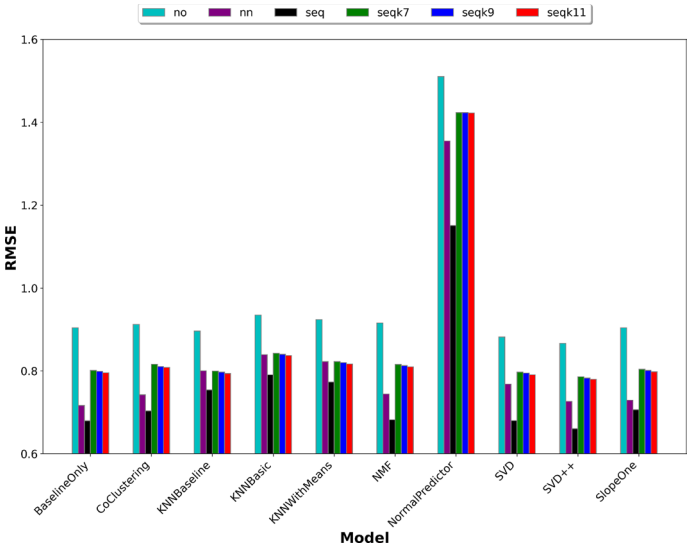

Fig. 7 RMSE results for the multiple models and natural noise approaches selected in the MovieLens last 1 M of 25 M ratings dataset

Table 9 Percentage of improvement (%) of the *seq* proposal versus the state-of-the-art *nn* proposal for MovieLens 1 M dataset

Models	RMSE (%)	MAE (%)	NDCG (%)	Precision (%)	Recall (%)	F1 (%)	Modified Ratings (%)	Running Time (secs) (%)
Baseline-Only	5.15211	5.95850	0.41497	0.24633	1.79250	1.09647	−29.05052	−2389.04875
CoClustering	5.20821	6.58915	0.44559	−0.18068	0.56532	0.23268	−41.47016	−2723.08614
KNNBaseline	5.78982	6.23572	0.56657	0.54577	1.64881	1.15671	−59.96698	−2540.95459
KNNBasic	5.80389	6.28775	0.58843	0.65155	1.59848	1.16481	−70.58906	−2276.45257
KNNWithMeans	6.01957	6.70524	0.55562	0.25032	1.25030	0.81044	−55.69603	−2385.20671
NMF	8.37040	10.28306	0.94358	−0.03900	0.89545	0.48613	−60.36098	−2724.94092
NormalPredictor	15.08345	16.24543	3.94665	6.14410	4.78837	5.35020	−59.06416	−2515.47151
SVD	11.50817	13.32883	1.05505	0.02031	1.14400	0.64723	−124.87411	−2604.38175
SVD++	9.06901	10.83952	0.72816	−0.20894	0.93739	0.43033	−79.83657	−2100.32074
SlopeOne	3.19498	3.44131	0.25489	0.11910	0.92239	0.56805	−17.28566	−2686.13991

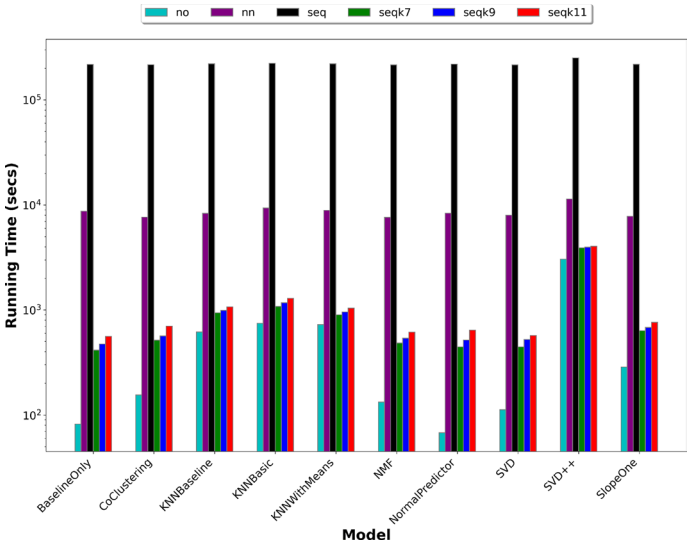

Fig. 8 Running time (secs) results, in logarithmic scale, for the multiple models and natural noise approaches selected in the MovieLens last 1 M of 25 M ratings dataset

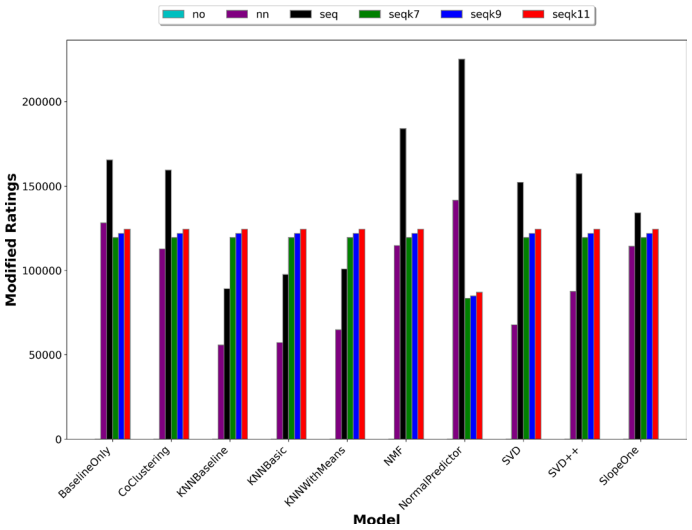

Fig. 9 Number of modified ratings produced for each model and natural noise approach evaluated combination in the MovieLens last 1 M of 25 M ratings dataset

Additionally, Table 8 shows an increase in the running time for the *seq* approach, while the *seqk* approach shows a significant reduction in the time cost. Because each time window is seven days, the *seq* approach can be applied in a

real-world application without a problem. In the case of time constraints, such as to prevent the use of the *seq* or *nn* approaches, the *seqk* approaches offer competitive results (Fig. 7), with respect to the traditional approach (*nn*). Moreover, this approach allows us to improve its performance in terms of results by adapting the time horizon k to the time constraints of each problem.

As in the previous case, the comparison between the *nn* model and the *seq* model, by percentage of improvement, is included in Table 9. The results show that a significant enhancement is evident across nearly all performance metrics. When examining factors such as running time and the number of modified values, it becomes clear that the *seq* approach is more time-consuming and invasive. Both metrics highlight the trade-off involved in achieving improved results.

Analyzing the results obtained by all models and approaches in both datasets, we can see that, except for the KNNBaseline model, the rest of the models obtain, in most cases, an improvement in the performance of the results. If we focus on the *seqk* approaches, we can appreciate that the performance differences for different values of k are more significant in the case of MovieLens100k than in MovieLens1M (Figs. 4 and 7, respectively). These results show the importance of the temporal component in both scenarios, but it is more significant in small datasets. In addition, the importance of the parameter k can be appreciated, and it is advisable to adapt it to the type of problem addressed.

The results obtained for the *seq* approach, which are the best results in the vast majority of cases, require a large amount of running time (Fig. 8). Because, in a real case, the accumulation of data takes weeks or even months, for the amount of data we are working on, the running time does not limit the application of our proposal in real scenarios. In the extreme case of working with large amounts of data and very tight model running time windows, we always have the option of using the *seqk* approach. This approach allows us to adjust the performance and running time using the parameter k, which is especially useful for this type of problem.

Finally, in Fig. 9, we analyze the intrusivity levels of our MovieLens1M dataset and compare it with those obtained for the MovieLens100k dataset, Fig. 6, we can appreciate a significant reduction of the relative intrusivity in each dataset. This behavior can be observed numerically by comparing the results included in Tables 6 and 8. This is especially relevant in the case of the KNN-based models, where the *seq* approach is the second least intrusive, marking a significant difference from the trend shown by the rest of the cases in both datasets.

4.5 Discussion

The results obtained in the previous section show a considerable improvement in the data quality after the application of the two natural noise correction techniques proposed in this study.

The first approach proposed, *seq*, obtains considerable results improvement by adding and accumulating information sequentially (Tables 6 and 8). Although the intrusiveness is not high according to Figs. 6 and 9, the computation required in high-dimensional problems may limit its use.

The second proposed approach, *seqk*, focused on the use of data related to the last k weeks, is able to provide competitive results in a short time at the cost of higher intrusiveness, considering the traditional natural noise approach (Figs. 6 and 9) and a correct setting of the parameter k. This proposal uses a smaller amount of data, which allows its use in real problems with large dimensions and running time limitations (Figs. 5 and 8).

Based on the obtained results, the application of natural noise correction techniques has shown a robust increase in the quality of the processed data. For this reason, its use is highly recommended in any RS problem. Furthermore, with the development of the current work, it has been proved that applying natural noise management approaches over small segments of rating data in RS is feasible, which is a step-further viewpoint in relation to the former works in natural noise management [13, 86], which have always used the whole dataset as input. According to our viewpoint, this is one of the main contributions of this work in relation to previous contributions.

In addition, a well-defined balance was observed between the volume of data used for training the natural noise management model and the degree of accuracy improvement linked to such a trained model. Thus, a larger number of rating segments used for training natural noise management leads to a larger accuracy improvement. However, the use of a lower number of rating sequences in the correction implies a more modest accuracy improvement. Nevertheless, in this case, it is important to note that fewer sequences imply a lower running time, which could be a variable that must be controlled in practical application scenarios of the methods discussed.

The proposals presented in this paper improve the quality of the data processed in recommender systems by incorporating temporal information of interest into the natural noise-cleaning process in a transparent way for the end user. This is because the two proposals can be applied to any recommender system with temporal information since they can be included as an additional step just before introducing the data into the final model.

An important shortcoming related to the approaches presented in the current contribution is the lack of uncertainty management associated with the rating data. The management of uncertainty has been previously proven to be a useful component in natural noise management in recommender systems [85, 87]. Future work will focus on this direction. In addition, future work will explore different exponential functions for characterizing the importance of each rating, according to their associated timestamp, for building user profiles with the natural noise management-related task.

Furthermore, another important shortcoming of the current work is that it is specifically focused on individual recommendations. However, previous studies on natural noise management, such as [13], showed that this task has a very positive effect on group recommender systems. Therefore, these previous results highlight the necessity of exploring time-related natural noise management, as screened in the current work, in group recommender systems scenarios.

5 Conclusions

In recent years, several studies have shown that user preferences tend to be inconsistent, which affects the accuracy performance of recommender systems (RS). In order to address this issue, several preprocessing approaches have been developed, processing these anomalous behaviors and obtaining a positive impact on the recommendation precision.

In this study, we focus on the application of these preprocessing proposals in real-time RS. To this end, we propose two incremental strategies to correct noisy ratings in this scenario. Considering a simulated time-aware RS, we have shown how these strategies are appropriate considering the recommendation accuracy, running time, and intrusion level in the data. Specifically, it is important to highlight that the achieved recommendation accuracy outperforms the accuracy obtained by previous works on natural noise management, such as those discussed in Sect. 2, and that the identified intrusion degree is lower than the intrusion degree obtained by other data preprocessing tasks in related data mining scenarios [6].

Beyond the theoretical and experimental results obtained across this paper, the practical implications of the obtained results rely on the fact that they illustrate that the time-related, sequence-driven management of natural noise in recommender systems is feasible. While previous works in this area have focused on performing this task over a large batch of data, the current work illustrates that the noise correction of small data segments also leads to an improvement in prediction accuracy and could provide additional benefits such as smaller intrusiveness and a shorter running time. This natural noise management over small data segments could be the key for generalizing these types of approaches in currently deployed recommendation applications, considering their huge amount of data that makes the use of previous methods focused on the entire dataset inappropriate. This work holds potential applications in domains characterized by the continuous generation of content, which often experiences brief periods of trending. It requires proposals capable of effectively integrating the temporal information and enhancing the data quality for the final model. Examples of such domains include streaming platforms and social networks, which frequently produce trending content within specific timeframes.

In the next future work, the current proposals will be extended to the group recommendation scenario [13]. Furthermore, we will focus on reformulating the current proposals using fuzzy tools [14, 85]. As a major goal, we aim to minimize the number of corrections required in past ratings and eventually work toward a framework in which correction is performed just at the rating insertion moment. For this purpose, we intend to exploit the sequential pattern mining theory [52] to model, at least partially, the inconsistencies that appear.

Additionally, we pretend to validate the current proposal through its use for recommendation improvement in practical cases such as e-learning scenarios [89]. Finally, explainable recommendations should also be considered in this environment [82].

Appendix A

Algorithm A.1 Method to identify possible noisy ratings

Algorithm A.1: Method to identify possible noisy ratings

Input : R_{data} the sorted set of ratings to be checked for natural noise detection
r_{ui} current rating
$th1$ first classification thresholds
$th2$ second classification thresholds

Output: R_{noise} list of ratings identified as noise

identifyNoiseRatings(R_{data}, $th1$, $th2$)

```
1  R_noise = { }
2  U={users providing ratings in R_data}
3  I={items rated in R_data}
4  ClassifyUsers(U)
5  ClassifyItems(I)
6  foreach r_ui in R_data do
7  |  if u is critical and i is weakly-preferred and r_ui ≥ th1 then
8  |  |  R_noise=R_noise+r_ui
9  |  if u is average and i is averagely-preferred and (r_ui < th1 or r_ui
   |   ≥ th2) then
10 |  |  R_noise=R_noise+r_ui
11 |  if u is benevolent and i is strongly-preferred and r_ui < th2 then
12 |  |  R_noise=R_noise+r_ui
13 end foreach
14 return R_noise
```

Algorithm A.2 The rating correction approach

Algorithm A.2: The rating correction approach

Input : R_{data} the sorted set of ratings to be checked for natural noise detection
r_{ui} current rating to correct
δ difference threshold

Output: r_{ui} rating corrected

correctNoisyRating(r_{ui}, R_{data}, δ)

```
1  n_ui=predict(u,i,R)
2  if abs(r_ui-n_ui)>δ then
3  |  r_ui=n_ui
```

Acknowledgements This research is supported by the Plan Andaluz de Investigación, Desarrollo e Innovación (PAIDI 2020) under the project PROYEXCEL_00257. F.J. Baldán was supported by the Spanish Government Juan de la Cierva Formación contract (FJC2021-047112-I). R. Yera was supported by

a Grant for the Requalification of the Spanish University System for 2021–2023 in the María Zambrano modality (UJAR10MZ).

Author contributions FJ Baldán contributed to conceptualization, methodology, investigation, data curation, visualization, software, writing-original draft, Validation. RY contributed to conceptualization, methodology, writing-original draft, Validation. LM contributed to writing-review & editing, validation.

Funding Funding for open access publishing: Universidad de Jaén/CBUA.

Data availability All the used data were taken from public datasets available for the research community.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Abdalla HI, Amer AA, Amer YA, Nguyen L, Al-Maqaleh B (2023) Boosting the item-based collaborative filtering model with novel similarity measures. Int J Comput Intell Syst 16:123
2. Acharya A, Singh B, Onoe N (2023) Llm based generation of item-description for recommendation system. In: Proceedings of the 17th ACM Conference on Recommender Systems, pp. 1204–1207
3. Adomavicius G, Bauman K, Tuzhilin A, Unger M (2022) Context-aware recommender systems: from foundations to recent developments. In: Recommender systems handbook. Springer, pp. 211–250
4. Adomavicius G, Tuzhilin AT (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17:734–749
5. Alabduljabbar R, Alshareef M, Alshareef N (2023) Time-aware recommender systems: a comprehensive survey and quantitative assessment of literature. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3274117
6. Alexandropoulos S-AN, Kotsiantis SB, Vrahatis MN (2019) Data preprocessing in predictive data mining. Knowl Eng Rev 34:e1
7. Amatriain X, Jaimes* A, Oliver N, Pujol JM (2010) Data mining methods for recommender systems. In: Recommender systems handbook. Springer, pp. 39–71
8. Amatriain X, Pujol JM, Tintarev N, Oliver N (2009) Rate it again: increasing recommendation accuracy by user re-rating. In: Third ACM Conference on Recommender Systems. ACM, pp. 173–180
9. Bag S, Kumar S, Awasthi A, Tiwari MK (2019) A noise correction-based approach to support a recommender system in a highly sparse rating environment. Decis Support Syst 118:46–57
10. Bellogín A, Said A, de Vries AP (2014) The magic barrier of recommender systems–no magic, just ratings. In: International Conference on User Modeling, Adaptation, and Personalization. Springer, pp. 25–36
11. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl-Based Syst 46:109–132
12. Campos PG, Díez F, Cantador I (2014) Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model User-Adap Inter 24:67–119

13. Castro J, Yera R, Martínez L (2017) An empirical study of natural noise management in group recommendation systems. Decis Support Syst 94:1–11
14. Castro J, Yera R, Martínez L (2018) A fuzzy approach for natural noise management in group recommender systems. Expert Syst Appl 94:237–249
15. Chen Y, Liu Z, Li J, McAuley J, Xiong C (2022) Intent contrastive learning for sequential recommendation. In: Proceedings of the ACM Web Conference 2022, pp. 2172–2182
16. Chen Y-C, Hui L, Thaipisutikul T (2021) A collaborative filtering recommendation system with dynamic time decay. J Supercomput 77:244–262
17. Chen Y-L, Yeh Y-H, Ma M-R (2021) A movie recommendation method based on users' positive and negative profiles. Inf. Process. Manag. 58:102531
18. Dacrema MF, Cantador I, Fernández-Tobías I, Berkovsky S, Cremonesi P (2022) Design and evaluation of cross-domain recommender systems. In: Recommender systems handbook. Springer, pp. 485–516
19. De Gemmis M, Lops P, Musto C, Narducci F, Semeraro G (2015) Semantics-aware content-based recommender systems. In: Recommender systems handbook, pp. 119–159
20. De Pessemier T, Dooms S, Martens L (2014) Comparison of group recommendation algorithms. Multimed Tools Appl 72:2497–2541
21. Deldjoo Y, Dacrema MF, Constantin MG, Eghbal-Zadeh H, Cereda S, Schedl M, Ionescu B, Cremonesi P (2019) Movie genome: alleviating new item cold start in movie recommendation. User Model User-Adap Inter 29:291–343
22. Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22:143–177
23. Ding Y, Li X (2005) Time weight collaborative filtering. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management. ACM, pp. 485–492
24. Ekstrand MD, Riedl JT, Konstan JA (2011) Collaborative filtering recommender systems. Found. Trends Hum.-Comput. Interact. 4:81–173
25. George T, Merugu S (2005) A scalable collaborative filtering framework based on co-clustering. In: Fifth IEEE International Conference on Data Mining (ICDM'05). IEEE, pp. 4
26. Goyani M, Chaurasiya N (2020) A review of movie recommendation system: limitations, survey and challenges. ELCVIA Electron Lett Comput Vis Image Anal 19:18–37
27. Gunawardana A, Shani G (2009) A survey of accuracy evaluation metrics of recommendation tasks. J Mach Learn Res 10:2935–2962
28. Gunes I, Kaleli C, Bilge A, Polat H (2014) Shilling attacks against recommender systems: a comprehensive survey. Artif Intell Rev 42:767–799
29. Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst (TIIS) 5:1–19
30. Huang X, Fang Q, Qian S, Sang J, Li Y, Xu C (2019) Explainable interaction-driven user modeling over knowledge graph for sequential recommendation. In: Proceedings of the 27th ACM International Conference on Multimedia, pp. 548–556
31. Hug N (2020) Surprise: a python library for recommender systems. J Open Sour Softw 5:2174
32. Jannach D, Manzoor A, Cai W, Chen L (2021) A survey on conversational recommender systems. ACM Comput Surv (CSUR) 54:1–36
33. Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) 20:422–446
34. Jin Z, Zhang Y, Mu W, Wang W, Jin H (2018) Leveraging the dynamic changes from items to improve recommendation. In: Conceptual Modeling: 37th International Conference, ER 2018, Xi'an, China, October 22–25, 2018, Proceedings 37. Springer, pp. 507–520
35. Kluver D, Nguyen TT, Ekstrand M, Sen S, Riedl J (2012) How many bits per rating? In: Proceedings of the Sixth ACM Conference on Recommender Systems, pp. 99–106
36. Konstan JA, Riedl J (2012) Recommender systems: from algorithms to user experience. User Model User-Adap Inter 22:101–123
37. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434
38. Koren Y (2010) Collaborative filtering with temporal dynamics. Interact Comput ACM 53:89–97
39. Koren Y (2010) Factor in the neighbors: scalable and accurate collaborative filtering. ACM Trans Knowl Discov Data (TKDD) 4:1–24

40. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42:30–37
41. Kumar S, De K, Roy PP (2020) Movie recommendation system using sentiment analysis from microblogging data. IEEE Trans Comput Soc Syst 7:915–923
42. Latrech J, Kodia Z, Ben Azzouna N (2023) CoDFI-DL: a hybrid recommender system combining enhanced collaborative and demographic filtering based on deep learning. J Supercomput. https://doi.org/10.1007/s11227-023-05519-2
43. Lemire D, Maclachlan A (2005) Slope one predictors for online rating-based collaborative filtering. In: Proceedings of the 2005 SIAM International Conference on Data Mining. SIAM, pp. 471–475
44. Li B, Chen L, Zhu X, Zhang C (2013) Noisy but non-malicious user detection in social recommender systems. World Wide Web 16:677–699
45. Li X, Barajas JM, Ding Y (2007) Collaborative filtering on streaming data with interest-drifting. Intell Data Anal 11:75–87
46. Lops P, Jannach D, Musto C, Bogers T, Koolen M (2019) Trends in content-based recommendation: preface to the special issue on recommender systems based on rich item descriptions. User Model User-Adap Inter 29:239–249
47. Luo C, Wang Y, Li B, Liu H, Wang P, Zhang LY (2023) An efficient approach to manage natural noises in recommender systems. Algorithms 16:228
48. Luo X, Xia Y, Zhu Q, Li Y (2013) Boosting the k-nearest-neighborhood based incremental collaborative filtering. Knowl-Based Syst 53:90–99
49. Luo X, Zhou M, Xia Y, Zhu Q (2014) An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans Industr Inf 10:1273–1284
50. Martínez L, Castro J, Yera R (2016) Managing natural noise in recommender systems. In: Martín-Vide C, Mizuki T, Vega-Rodríguez MA (eds) Theory and Practice of Natural Computing: 5th International Conference, TPNC 2016, Sendai, Japan, December 12–13, 2016, Proceedings. Springer International Publishing, pp 3–17
51. Masthoff J, Delić A (2022) Group recommender systems: beyond preference aggregation. In: Recommender Systems Handbook. Springer, pp. 381–420
52. Mishra R, Kumar P, Bhasker B (2015) A web recommendation system considering sequential information. Decis Support Syst 75:1–10
53. Mobasher B, Burke R, Bhaumik R, Williams C (2007) Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness. ACM Trans Internet Technol. https://doi.org/10.1145/1278366.1278372
54. Mohammadpour T, Bidgoli AM, Enayatifar R, Seyyed Javadi Haj H (2023) Efficient recommendations in collaborative filtering recommender system: a multi-objective evolutionary approach based on nsga-ii algorithm. Int J Nonlinear Anal Appl 14:785–804
55. Ning X, Desrosiers C, Karypis G (2015) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B (eds) Recommender systems handbook. Springer, US, pp 37–76
56. O'Mahony MP, Hurley NJ, Silvestre G (2006) Detecting noise in recommender system databases. In: 11th International Conference on Intelligent User Interfaces. ACM, pp. 109–115
57. Park H, Jeong J, Oh K-W, Kim H (2023) Autoencoder-based recommender system exploiting natural noise removal. IEEE Access 11:30609–30618
58. Pérez-Almaguer Y, Yera R, Alzahrani AA, Martínez L (2021) Content-based group recommender systems: a general taxonomy and further improvements. Expert Syst Appl 184:115444
59. Pham HX, Jung JJ (2013) Preference-based user rating correction process for interactive recommendation systems. Multimed Tools Appl 65:119–132
60. Pilászy I, Tikk D (2009) Recommending new movies: even a few ratings are more valuable than metadata. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 93–100
61. Quadrana M, Cremonesi P, Jannach D (2018) Sequence-aware recommender systems. ACM Comput Surv (CSUR) 51:1–36
62. Rabiu I, Salim N, Da'u A, Osman A (2020) Recommender system based on temporal models: a systematic review. Appl Sci 10:2204
63. Rabiu I, Salim N, Da'u A, Osman A (2020) Recommender system based on temporal models: a systematic review. Appl Sci 10:2204
64. Rendle S (2022) Item recommendation from implicit feedback. In: Recommender Systems Handbook. Springer, pp. 143–171

65. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work. ACM, New York, pp. 175–186
66. Resnick P, Varian HR (1997) Recommender systems. Commun ACM 40:56–58
67. Ricci F, Venturini A, Cavada D, Mirzadeh N, Blaas D, Nones M (2003) Product recommendation with interactive query management and twofold similarity. In: International Conference on Case-Based Reasoning. Springer, pp. 479–493
68. Saia R, Boratto L, Carta S (2016) A semantic approach to remove incoherent items from a user profile and improve the accuracy of a recommender system. J Intell Inf Syst 47:111–134
69. Said A, Jain BJ, Narr S, Plumbaum T (2012) Users and noise: The magic barrier of recommender systems. In: Masthoff J, Mobasher B, Desmarais MC, & Nkambou R (Eds.), User Modeling, Adaptation, and Personalization: 20th International Conference, UMAP 2012, Montreal, Canada, July 16-20, 2012. Proceedings. Springer Berlin Heidelberg, pp. 237–248
70. Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th international conference on Machine learning, pp. 880–887
71. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: 10th International Conference on World Wide Web. ACM, pp. 285–295
72. Tintarev N, Masthoff J (2022) Beyond explaining single item recommendations. In: Recommender Systems Handbook. Springer, pp. 711–756
73. Tran DT, Huh J-H (2023) New machine learning model based on the time factor for e-commerce recommendation systems. J Supercomput 79:6756–6801
74. Van Dat N, Van Toan P, Thanh TM (2022) Solving distribution problems in content-based recommendation system with gaussian mixture model. Appl Intell 52:1602–1614
75. Vinagre J, Jorge AM, Gama J (2015) An overview on the exploitation of time in collaborative filtering. Wiley Interdiscip Rev Data min Knowl Discov 5:195–215
76. Wang P, Wang Y, Zhang LY, Zhu H (2021) An effective and efficient fuzzy approach for managing natural noise in recommender systems. Inf Sci 570:623–637
77. Wang W, Mishra KK (2017) A novel stock trading prediction and recommendation system. In: Multimedia Tools and Applications, pp. 1–13
78. Wang Y, Wang L, Li Y, He D, Liu T-Y (2013) A theoretical analysis of NDCG type ranking measures. In: Conference on Learning Theory. PMLR, pp. 25–54
79. Wei J, He J, Chen K, Zhou Y, Tang Z (2017) Collaborative filtering and deep learning based recommendation system for cold start items. Expert Syst Appl 69:29–39
80. Widiyaningtyas T, Hidayah I, Adji TB (2021) User profile correlation-based similarity (UPCSIM) algorithm in movie recommendation system. J Big Data 8:1–21
81. Yannam VR, Kumar J, Babu KS, Patra BK (2023) Enhancing the accuracy of group recommendation using slope one. J Supercomput 79:499–540
82. Yera R, Alzahrani AA, Martínez L (2022) Exploring post-hoc agnostic models for explainable cooking recipe recommendations. Knowl-Based Syst 251:109216
83. Yera R, Alzahrani AA, Martínez L (2022) A fuzzy content-based group recommender system with dynamic selection of the aggregation functions. Int J Approx Reason 150:273–296
84. Yera R, Alzahrani AA, Martínez L, Rodríguez RM (2023) A systematic review on food recommender systems for diabetic patients. Int J Environ Res Public Health 20:4248
85. Yera R, Barranco MJ, Alzahrani AA, Martínez L (2019) Exploring fuzzy rating regularities for managing natural noise in collaborative recommendation. Int J Comput Intell Syst 12:1382–1392
86. Yera R, Caballero Mota Y, Martínez L (2015) Correcting noisy ratings in collaborative recommender systems. Knowl-Based Syst 76:96–108
87. Yera R, Castro J, Martínez L (2016) A fuzzy model for managing natural noise in recommender systems. Appl Soft Comput 40:187–198
88. Yera R, Martínez L (2017) Fuzzy tools in recommender systems: a survey. Int J Comput Intell Syst 10:776–803
89. Yera R, Martínez L (2017) A recommendation approach for programming online judges supported by data preprocessing techniques. Appl Intell 47:277–290
90. Yu L, Han F, Huang S, Luo Y (2017) A content-based goods image recommendation system. Multimed Tools Appl. https://doi.org/10.1007/s11042-017-4542-z
91. Yu P, Lin L, Yao Y (2016) A novel framework to process the quantity and quality of user behavior data in recommender systems. In: Cui B, Zhang N, Xu J, Lian X, Liu D (eds) Web-Age Information

Management: 17th International Conference, WAIM 2016, Nanchang, China, June 3–5, 2016, Proceedings, Part I. Springer International Publishing, pp 231–243
92. Zhang S, Tay Y, Yao L, Sun A, Zhang C (2022) Deep learning for recommender systems. In: Recommender Systems Handbook. Springer, pp. 173–210
93. Zhou K, Wang H, Zhao WX, Zhu Y, Wang S, Zhang F, Wang Z, Wen J-R (2020) S3-rec: self-supervised learning for sequential recommendation with mutual information maximization. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1893–1902
94. Zhu J, Han L, Gou Z, Yuan X (2018) A fuzzy clustering-based denoising model for evaluating uncertainty in collaborative filtering recommender systems. J Am Soc Inf Sci 69:1109–1121

Authors and Affiliations

Francisco J. Baldán[1,2] · Raciel Yera[2] · Luis Martínez[2]

✉ Raciel Yera
ryera@ujaen.es

Francisco J. Baldán
fjbaldan@uma.es

Luis Martínez
martin@ujaen.es

[1] Department of Computer Science and Programming Languages, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Málaga, 29071 Málaga, Spain

[2] Computer Science Department, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Jaén, 23071 Jaén, Spain