SN Computer Science (2021) 2:255
https://doi.org/10.1007/s42979-021-00668-8

ORIGINAL RESEARCH

On Addressing the Low Rating Prediction Coverage in Sparse Datasets Using Virtual Ratings

Dionisis Margaris[1] · Dimitris Spiliotopoulos[2] · Gregory Karagiorgos[3] · Costas Vassilakis[3] · Dionysios Vasilopoulos[3]

Received: 14 February 2021 / Accepted: 26 April 2021

Abstract
Collaborative filtering-based recommendation systems consider users' likings and interests, articulated as ratings within a database to offer personalized recommendations. Unfortunately, many collaborative filtering datasets exhibit the "grey sheep" phenomenon, a state where no near neighbours can be found for certain users. This phenomenon is extremely frequent in datasets where users, on average, have rated only a small percentage of the available items, which are termed as sparse datasets. This paper addresses the "grey sheep" problem by proposing the virtual ratings concept and introduces an algorithm for virtual rating creation on the basis of actual ratings. The novelty behind this concept is that the introduction of the virtual ratings effectively reduces the user–item rating matrix sparsity, thus alleviating the aforementioned problem. The proposed algorithm, which is termed as CF_{VR}, has been extensively evaluated and the results show that it achieves to considerably improve the capability of a collaborative filtering system to formulate tailored recommendations for each user, when operating on sparse datasets, while at the same time improves rating prediction quality.

Keywords Recommender systems · Collaborative filtering · Virtual ratings · Sparse datasets · Pearson correlation coefficient · Cosine similarity · Evaluation

Introduction

Collaborative filtering (CF)-based recommendation systems consider users' likings and interests, articulated as ratings within a database to offer personalized recommendations. CF algorithms are categorized either as user–user (or user-based) or as item–item (item-based). User–user CF algorithms, which is the area addressed in this paper, initially create each user's *U* "near neighbourhood", i.e. a set of users whose ratings are similar to *U*'s ratings. Then, the ratings of *U*'s near neighbours (NNs) are used to create rating predictions for *U*, and finally, these rating predictions drive the recommendation formulation process [1].

Costas Vassilakis
costas@uop.gr

[1] Department of Digital Systems, University of the Peloponnese, Sparti, Greece

[2] Department of Management Science and Technology, University of the Peloponnese, Tripoli, Greece

[3] Department of Informatics and Telecommunications, University of the Peloponnese, Tripoli, Greece

However, many CF datasets suffer from the "grey sheep" problem, a state where no NNs can be found for certain users. The aforementioned issue is more acute in sparse datasets, that is datasets where users—on average—have rated only a small percentage of the available items [2].

Previous works have addressed the aforementioned problem, by incorporating the concept of virtual near neighbours (VNNs): a VNN is an artificially generated user entity that combines the ratings of real users and the enrichment of the user rating database with VNNs leverages the CF prediction generation process and increasing coverage. VNNs are introduced and used in the CF_{VNN} algorithm [3].

In this paper, we address the aforementioned problem by introducing the novel concept of virtual ratings (VRs), as well as an associated algorithm that processes actual (existing) ratings and generates VRs. Virtual ratings are rating predictions that are added into the user–item rating database. Once generated, VRs are used into the rating prediction computation process, in a fashion similar to that of user-contributed ratings. The novelty of the proposed algorithm the exploitation of the direct and the indirect neighbourhood of the user, up to a radius, for the generation of a comprehensive set of VRs, achieving thus a considerable reduction of the rating matrix

Published online: 06 May 2021

sparsity. Furthermore, the algorithm introduced in this paper acknowledges that VRs are essentially rating predictions, and therefore, bear a degree of uncertainty: to this end, the algorithm proposed in this paper computes a weight associated with each VR, which reflects the degree of confidence to the value of the VR, and this weight is taken into account in the rating prediction formulation procedure.

To exemplify the concept of VRs, let us examine the case of computing a prediction for rating r_{U_1,i_1} that user U_1 would give to item i_1, so as to determine whether it is suitable for inclusion in a recommendation to U_1. In this illustrative example, we assume that the recommender system (RS) employs the Pearson correlation coefficient (PCC) metric to compute user-to-user similarity [1, 2] and the current state of the user rating database is as shown in Table 1.

In this setting, U_2 is the only NN of U_1 (their PCC score is positive, because the only item that they have rated in common is i_2 and each user's rating for i_2 is below the same user's average rating), however, U_2 has not rated i_1, and, therefore, it is not possible to formulate a prediction $p\left(r_{U_1,i_1}\right)$ for the rating r_{U_1,i_1}. And while user U_3 has rated i_1, r_{U_3,i_1} cannot be used to compute $p\left(r_{U_1,i_1}\right)$, because U_3 does not belong to U_1's NN set (users U_1 and U_3 have relatively opposite ratings on their commonly rated items i_5, i_6 and i_7, leading to a low similarity value).

The VR algorithm proposed in this paper recognizes that the near neighbours of U_1 cannot contribute to the prediction for the rating r_{U_1,i_1} and triggers a process for computing a rating prediction for item i_1 for each of U_1's NNs. In our example, U_3 is a NN to U_2 (i.e. the only NN of U_1), because the only item rated in common by U_2 and U_3 is i_4, and both U_2 and U_3 have rated i_4 with a high mark, and U_3 has rated i_1, hence a rating prediction $p\left(r_{U_2,i_1}\right)$ for r_{U_2,i_1} can be generated. Once generated, $p\left(r_{U_2,i_1}\right)$ will be converted to a VR vr_{U_2,i_1} and accommodated into the user–item rating matrix, enabling the computation of a rating prediction $p\left(r_{U_1,i_1}\right)$.

To substantiate the performance of the $\mathrm{CF}_{\mathrm{VR}}$ algorithm, we have performed an extensive experimental validation exploring (a) the gains in terms of prediction coverage increase attained by the $\mathrm{CF}_{\mathrm{VR}}$ algorithm and (b) how the $\mathrm{CF}_{\mathrm{VR}}$ algorithm affects the quality of rating predictions. In this experimental validation, we utilized eight contemporary and widely used datasets covering a variety of domains (movies, books, food, music, etc.), and we take into account two broadly employed similarity metrics for CF systems, namely the Pearson correlation coefficient (PCC) and the cosine similarity (CS) [1, 2]. We also comparatively assess the performance of the proposed $\mathrm{CF}_{\mathrm{VR}}$ algorithm to that of the $\mathrm{CF}_{\mathrm{VNN}}$ algorithm, proposed in [3] as well as the $\mathrm{CF}_{\mathrm{DR}}$ algorithm proposed in [4]; both $\mathrm{CF}_{\mathrm{VNN}}$ and $\mathrm{CF}_{\mathrm{DR}}$ are state-of-the-art algorithms that also target prediction coverage improvement in CF over sparse datasets.

Considering the above, the novel aspects of this work are as follows:

- we introduce the concept of VRs, i.e. virtual ratings, where a virtual rating $\mathrm{vr}_{U,i}$ is computed when rating $r_{U,i}$ is not present in the user–item rating matrix, but its presence would enable the formulation of a rating prediction on item i for some user V that is a NN of U
- we explore how the created VRs can be incorporated into the rating prediction formulation process, targeting primarily to increase the prediction coverage of CF-based recommender systems (RSs), while ensuring that prediction accuracy is at least maintained at the same levels. The inclusion of the VR concept into the CF technique synthesises a novel CF-based algorithm, which will be denoted as $\mathrm{CF}_{\mathrm{VR}}$.
- We present an extensive experimental evaluation of the proposed algorithm, demonstrating that the proposed algorithm introduces considerable gains in terms of coverage, while in parallel improving rating prediction quality. The proposed algorithm is also shown to surpass the performance of other state-of-the-art algorithms.

The proposed algorithm can be directly used in the implementation of efficient recommender systems, while it can also leverage further research: first, it can be easily fused with further algorithms which focus on (a) the improvement of the rating prediction computation performance, (b) the increase of the accuracy of rating prediction or (c) the improvement of recommendation quality in CF-based RSs. These algorithms include matrix factorization techniques [5], exploitation of visual information [6], pruning of old user ratings [7, 8], concept drift detection techniques [9–11], clustering techniques [12–14], algorithms utilizing data from social networks (e.g. user relationship graphs) [15–17], or algorithms applying hybrid filtering [18–21].

The rest of the paper is structured as follows: "Related work" overviews related work, while "The proposed algorithm" introduces the $\mathrm{CF}_{\mathrm{VR}}$ algorithm. "Experimental evaluation" elaborates on the procedure for tuning parameter values used within the operation algorithm and the evaluation of the proposed technique, and reports on the evaluation of the algorithm; in both tasks, eight contemporary and widely used CF datasets were utilized. Finally, "Conclusions and future work" concludes the paper and outlines future work.

Table 1 CF rating database example

User/item	i_1	i_2	i_3	i_4	i_5	i_6	i_7
U_1	?	1	2		5	5	4
U_2		2		5			
U_3	5			5	1	1	2

Related Work

CF-based systems have attracted substantial research focus over the last years, due to their utility in a number of real-world tasks and applications. However, most of these works aim to improve rating prediction accuracy and recommendation quality, while the aspect prediction/recommendation coverage has not been equally developed [22, 23].

Vozalis et al. [18] introduce ItemHyCov, a CF-based algorithm that fuses the merits of user-based and item-based CF approaches into a hybrid CF approach that applies feature combination, aiming to improve the coverage levels of CF rating predictions over sparse datasets. Margaris and Vassilakis [24] present the so-called $\mathrm{CF}_{\mathrm{negNNs}}$ algorithm, which increases the coverage attainable by CF-based algorithms in sparse datasets, by considering in the rating prediction computation phase users that bear a high degree of dissimilarity to the user for whom each prediction is computed.

Pham et al. [14] present a CF recommendation method based on clustering. In this approach, neighbourhoods are computed on the basis of social network relationships between users, rather than rating similarity. In this sense, the algorithm proposed in [14] utilizes a clustering technique for complex networks to process the social network user graph and thus generate user clusters, with the users in each cluster being considered as near neighbours. Wang et al. [25] utilize user trust relationships to leverage traditional CF-based techniques. Trust relationships alleviate the issues introduced due to data sparsity and cold-start problems, allowing the formation of near neighbourhoods even in the absence of an adequate number of co-rated items. The work of Zarei and Moosavi [26] introduces a CF algorithm based on a social memory and explore the consideration of social ties in the recommendation process, and the impact of this addition on rating prediction accuracy and coverage. Margaris et al. [16] propose an algorithm for SN-based recommender systems, where the density of the CF-based and the SN-based neighbourhoods fluctuate significantly across different users. This algorithm initially computes one prediction for the SN neighbourhood and one prediction for the CF-based neighbourhood, and then combines the two distinct predictions via a weighted sum approach to formulate the final prediction; the weight for each partial prediction is determined by taking into account the related neighbourhood (CF and SN) sizes. While all the aforementioned approaches improve CF coverage, this is made possible through the exploitation of supplementary information harvested from complementary sources, mainly social networks; however, this information is not always available, hence the applicability of these approaches is limited.

Matrix factorization (MF) techniques have emerged as a promising approach for the computation of rating predictions. The coverage problem is existent in MF-based systems, however, it exhibits a different manifestation: due to the nature of the rating prediction computation method employed in MF-based systems, a prediction for the rating $r_{U,i}$ that user U would assign to item i is always generated, nevertheless, as asserted in [8], predictions for users or items that have a very small number of ratings practically degenerate to some constant value that is dependent on the dataset, and effectively do not constitute personalized predictions [27]. In terms of metrics, coverage will be always equal to 100%, however, there is a negative impact on the rating prediction accuracy. Guan et al. [28] tackle this issue through the introduction of a singular value decomposition (SVD) model and utilization of active learning techniques, where a number of users are selected to be queried about a specific item, and their ratings are used to enhance the rating matrix and thus avoid the degeneration of predictions and improve rating prediction accuracy.

Poirier et al. [29] propose an approach which complements the user–item rating matrix using virtual ratings that are computed through the processing of textual reviews that are sourced from the social network. The accommodation of the text review-based virtual ratings decreases the sparsity of the user–item rating matrix, which has a positive effect on rating prediction coverage. An analogous methodology is followed by Moshfeghi et al. [30], where the emotions extracted from user reviews are used to estimate whether a user will like an item or not. The work in [31] presents an approach pertinent to the use of numeric ratings that are derived from textual reviews: in this approach, textual features are extracted from reviews and are used for computing a confidence factor for each textual review-based numeric rating, under the rationale that human language has an inherent degree of uncertainty. Although these approaches succeed to increase rating prediction coverage, they can only be applied when textual reviews are available. In contrast, the $\mathrm{CF}_{\mathrm{VR}}$ algorithm proposed in this paper operates using only the user–item rating matrix, not requiring any complementary data, and henceforth it can be applied to any CF dataset.

Recently, Margaris and Vassilakis [3] introduced the concept of artificial user profiles created by merging pairs of (real) NN profiles, namely virtual near neighbours (VNNs), and proposed an algorithm that incorporates the VNNs concept to alleviate the "grey sheep" problem. In the same line, the work in [32] exploits the "friend-of-a-friend" concept, where a user's near neighbourhood is transitively expanded, and this expansion of the near neighbourhood enables the computation of more rating predictions, increasing thus coverage. The respective algorithm is coined as $\mathrm{CF}_{\mathrm{FOAF.}}$ Additionally, the work in [4] proposes the $\mathrm{CF}_{\mathrm{DR}}$ algorithm which examines for each rating prediction (a) the number of users that have contributed to its computation, and (b) the cumulative similarity of these users and the user for which the prediction was computed and if both items meet some threshold, the rating prediction is considered as "robust" and inserted into the user–item rating matrix; in this way, the sparsity of the user–item rating matrix is lowered, leading to an increase in coverage. Notably, [4] shows that $\mathrm{CF}_{\mathrm{DR}}$ outperforms the $\mathrm{CF}_{\mathrm{FOAF}}$ algorithm.

Another major issue in sparse CF datasets is noise in the rating data that distort the accuracy of the CF systems [33–35]. Toledo et al. [36] propose an approach that detects and corrects the ratings that are deemed to be inconsistent and might introduce recommendation bias, based only on the user–item rating matrix and without requiring any complementary data. Yera et al. [37] develop a flexible approach to improve recommendation accuracy that uses fuzzy tools to provide a greater flexibility in the characterisation of the elements, which manages the uncertainty present in natural noise. Patra et al. [38] propose the Bhattacharyya CF Coefficient, which provides reliable item recommendations to users, by locating useful NNs in sparse CF dataset. Bag et al. [39] propose a method that first re-classifies the items and users of a RS into three classes (weak, average and strong) to identify and correct noise rating, and then predicts the values of the ratings of the unrated items, from the sparse and noise-free dataset.

The present work advances the state-of-the-art regarding the coverage increase in CF systems, through the reduction of the user–item rating matric sparsity by means of injection of the virtual ratings. Differently from other works, the injected virtual ratings are computed utilizing only the contents of the user–item rating matrix, not necessitating any additional data and henceforth being applicable to any CF-based system. Furthermore, while works [3, 32] that achieve coverage increase based only on the user–item rating matrix operate at user-level, the current work operates at individual rating level, being thus more versatile and achieving higher rating prediction coverage increase. In parallel, the injection of virtual ratings into the user–item rating matrix alleviates the "grey sheep" problem that all sparse CF datasets suffer from.

The Proposed Algorithm

CF algorithms operate in a two-phase fashion to compute predictions for the ratings that some user U would enter for items. The first phase encompasses the determination of U's NNs, i.e. the users who have rated items in a similar fashion with U. Rating similarity is quantified using a similarity metric, such as PCC or CS [1, 2]. Typically, only users whose similarity to U meets or exceeds a certain threshold are considered to be NNs for U. The set of U's NNs is denoted as $\text{NN}_{U.}$

The PCC is a commonly used metric for quantifying user-to-user similarity; under the PCC, the similarity between two users U and V is computed using formula (1):

$$\text{sim}_{\text{PCC}}(U,V)=\frac{\sum_k\left(r_{U,k}-\overline{r_U}\right)*\left(r_{V,k}-\overline{r_V}\right)}{\sqrt{\sum_k\left(r_{U,k}-\overline{r_u}\right)^2*\sum_k\left(r_{V,k}-\overline{r_V}\right)^2}},\quad(1)$$

where k iterates over the items that are rated by both U and V, while $\overline{r_U}$ (resp. $\overline{r_V}$) is the mean values of ratings entered in the database by U (resp. V).

CS [1, 2] is an alternative metric for quantifying user-to-user similarity and is expressed as:

$$\text{sim}_{\text{CS}}(U,V)=\frac{\sum_k r_{U,k}*r_{V,k}}{\sqrt{\sum_k\left(r_{U,k}\right)^2}*\sqrt{\sum_k\left(r_{V,k}\right)^2}}.\quad(2)$$

The second phase consists of the computation of rating predictions, which are personally tailored for U. To compute a prediction $p_{U,i}$ that U would enter for item i, formula (3) is employed:

$$p_{U,i}=\overline{r_u}+\frac{\sum_{V\in NN_u}\text{sim}(U,V)*\left(r_{V,i}-\overline{r_V}\right)}{\sum_{V\in NN_u}\text{sim}(U,V)}.\quad(3)$$

The novelty behind the proposed algorithm is the introduction of the virtual ratings (VRs), which are computed and used within the process of computing a rating prediction $p_{U,i}$ as follows: if no user $V\in NN_U$ has rated item i, and, therefore, a rating prediction for user U cannot be computed according to formula (3), then the procedure to generate virtual ratings $\text{vr}_{V,i}$ for each user $V\in\text{NN}_U$ is triggered. According to this procedure, for each user $V\in\text{NN}_U$ a rating prediction $p_{V,i}$ is calculated, based on the near neighbourhood of V. If, for at least one user $V\in\text{NN}_U$, a rating prediction $p_{V,i}$ was successfully calculated (i.e. the near neighbourhood of V included at least one NN W that had rated item i), then the VR calculation phase terminates, the calculated rating predictions are added to the user–item rating matrix as virtual ratings of the respective users and used in the process of the computation of rating prediction $p_{U,i}$ for user U. It is possible, however, that no user $V\in\text{NN}_U$ had a NN $W\in\text{NN}_V$ who had rated item i; in this case, the virtual rating calculation process is expanded recursively to all the near neighbourhoods of the members of NN_U, until either a virtual rating is computed, or no further users can be considered.

To exemplify the VR computation process, let us consider the CF NN network depicted in Fig. 1. In the context of this network, we need to compute a rating prediction $p_{U,i}$, however, none of the U's NNs has rated item i; only users W_2, W_3, X_2, X_3, Y_1 and Y_2 have rated item i.

Hence, the VR computation procedure is triggered as follows:

- Initially, V_1 is considered; V_1 has only one NN, W_1 (U obviously cannot serve as a recommender to V_1 for item i, since U is asking V_1 for a recommendation, and is, therefore, excluded). However, W_1 has not rated item i and has no near neighbours (other than V_1) to ask for a recommendation. Therefore, no VR is computed for V_1 regarding item i.
- Subsequently, V_2 is considered. V_2 has two direct NNs that have rated item i (W_2 and W_3), and therefore, $\text{vr}_{V_{2,i}}$ can be computed, by applying formula (3). User W_6 is also a NN of V_2, however, W_6 has not rated item i and hence does not contribute to the computation of $\text{vr}_{V_{2,i}}$.

- Finally, V_3 is considered. V_3 has two NNs (other than U), namely W_4 and W_5, none of which has rated item i, and consequently $\mathrm{vr}_{V_{3,i}}$ cannot be computed at this stage. However, users W_4 and W_5 have in turn near neighbours, and, therefore, the computation of $\mathrm{vr}_{W_{4,i}}$ and $\mathrm{vr}_{W_{5,i}}$ can be attempted on the basis of each user's NN. If at least one of these attempts is successful, then $\mathrm{vr}_{V_{3,i}}$ can be subsequently computed since the near neighbourhood of user V_3 would then include a user having a (virtual) rating for item i. Indeed, $\mathrm{vr}_{W_{4,i}}$ can be computed on the basis of the ratings of users X_2 and X_3, and $\mathrm{vr}_{W_{5,i}}$ can be computed based on the rating entered by user X_3; these VRs will then be combined using formula (3) to produce $\mathrm{vr}_{V_{3,i}}$. When computing $\mathrm{vr}_{W_{4,i}}$ users $\{V_3, U\}$ would be excluded from any consideration, since $\mathrm{vr}_{W_{4,i}}$ is being computed with the purpose of providing a rating prediction to them for item i (directly for V_3 and indirectly for U), and hence these users are not appropriate for providing a prediction back to W_4 in this process; similarly, users $\{V_3, U\}$ are excluded from consideration when computing $\mathrm{vr}_{W_{5,i}}$.
- Notably, the ratings entered by users Y_1 and Y_2 are not used in this process. The rating entered by user Y_1 is not used because user W_4 has direct NNs (reachable at a neighbourhood radius equal to 1) that have rated item i, and hence it is not necessary to consider indirect NNs (reachable at a neighbourhood radius greater than 1) for the computation of $\mathrm{vr}_{W_{4,i}}$; and the rating entered by Y_2 is not used, because Y_2 is not connected in U's CF neighbourhood, neither as a direct nor as an indirect NN.

Listing 1 presents the algorithm used to compute VRs. The code needed for excluding the user whose neighbourhood is being populated with VRs (directly or indirectly, as exemplified above) is omitted for clarity.

Since virtual ratings are effectively rating predictions, their values bear a degree of uncertainty. To accommodate this aspect in the rating prediction calculation process, a weight parameter is considered for each rating according to formula (4):

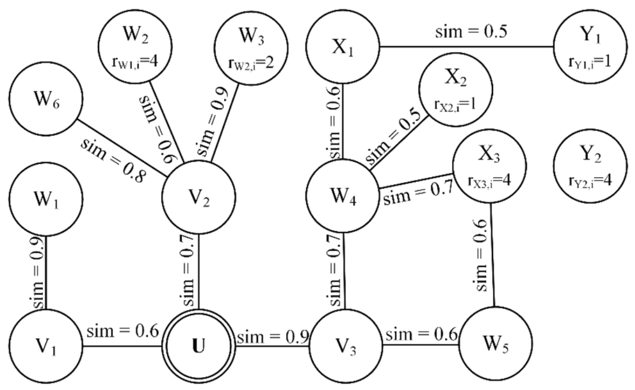

Fig. 1 A sample CF NN network

$$w_{\mathrm{rat}}(r) = \begin{cases} 1, & \text{if } r \text{ is an explicitly entered rating} \\ f(r) & \text{if } r = \mathrm{vr}_{V,i} \text{ is a virtual rating} \end{cases}, \tag{4}$$

where function f may take into account any property of VR r, and notably (1) the number of recursive steps taken to locate near neighbours that are able to contribute to the computation of r, which is denoted as radius(r), (2) the similarity of user V to his NNs that contribute their (virtual) ratings for the formulation of $\mathrm{vr}_{V,i}$ and (3) the weights of the (virtual) ratings that contribute to the computation of $\mathrm{vr}_{V,i.}$ The range of $f(r)$ is [0, 1]. In regard to the consideration of the radius(r) attribute, taking into account that a larger number of steps indicates a higher degree of uncertainty for the value of $\mathrm{vr}_{V,i}$, f will be a decreasing function with respect to radius(r).

Considering the introduction of rate weights, formula (3) is modified as shown in formula (5):

$$p_{U,i} = \overline{r_u} + \frac{\sum_{V\in \mathrm{NN}_U} \mathrm{sim}(U,V) * w_{\mathrm{rat}}\left(r_{V,i}\right) * \left(r_{V,i} - \overline{r_V}\right)}{\sum_{V\in \mathrm{NN}_U} \mathrm{sim}(U,V) * w_{\mathrm{rat}}\left(r_{V,i}\right)}, \tag{5}$$

where $r_{V,i}$ may be either a real rating (if one is available) or a virtual rating, computed according to the algorithm presented in Listing 1.

For the application of this algorithm, the following parameters need to be determined:

1. The number of recursive steps the VR computation algorithm needs to take in order to reach a satisfactory CF coverage level while maintaining efficiency in computing VRs; effectively, this number imposes a limit to the radius to which the near neighbourhood search will extend to when computing VRs, and will be denoted as $\mathrm{Rad}_{\mathrm{NN}}$ and
2. The optimal formulation for the f(radius) function computing w_{rat} in formula (4), i.e. the weights for the VRs taking part in the prediction formulation (w_{rat} for real ratings will be set to 1.0).

In "Experimental evaluation", we investigate different settings regarding the number of VR steps, as well as the values of the w_{rat} parameter, aiming to determine the optimal settings for the parameters of the proposed algorithm and finally evaluate the algorithm considering the dimensions of rating prediction coverage and accuracy.

Experimental Evaluation

In this section, we report on our experiments aiming to:

1. Determine the optimal values for the parameters required by the $\mathrm{CF}_{\mathrm{VR}}$ algorithm; these parameters are (a) the $\mathrm{Rad}_{\mathrm{NN}}$ limit and (b) the optimal weight set to

each virtual rating, based on the number of recursive steps that were taken to during its computation (i.e. the radius of the neighbourhood that was explored in the computation of the specific rating), and

2. Evaluate the CF_{VR} algorithm performance regarding the aspects of (a) coverage and (b) rating prediction accuracy. The CF_{VR} algorithm is comparatively evaluated to (1) the plain CF algorithm, which constitutes the baseline and (2) the CF_{VNN} algorithm [3] and the CF_{DR} algorithm [4].

```
FUNCTION computePrediction(User X, Item itm)
/*   Pseudocode for formula (3), used for the computation of VRs
     INPUT: X is the user for whom the VR will be computed; itm is the respective item
     OUTPUT: the rating prediction, or NULL if no NNs of X has rated itm, and hence a
             prediction cannot be computed */

     predictionNumerator = 0.0
     predictionDenominator = 0.0
     FOREACH Y ∈ $NN_X$
          IF ($r_{Y,itm}$ ≠ NULL) THEN
               predictionNumerator+= sim(X, Y) * ($r_{Y,i} - \overline{r_Y}$)
               predictionDenominator += sim(X, Y)
          ENDIF
     END /* FOREACH */
     IF (predictionDenominator = 0) THEN /* no NN of X has rated itm */
          RETURN NULL
     ELSE /* at least one NN of X has rated itm */
          return $\overline{r_X}$ + (predictionNumerator / predictionDenominator)
     END
END /* FUNCTION */

FUNCTION computeVirtualRating(User V, Item itm)
/* INPUT: V is the user for whom the VR will be computed; itm is the respective item.
   OUTPUT: the VR computed, and the number of recursive steps that had to be taken
           to compute the VR. If a VR cannot be computed, return NULL. */

/* If a rating prediction can be computed by V's NNs, return it, along with an indication
   that only the direct neighbourhood of V was used to compute the VR */
     VR_V_itm = computePrediction(V, itm)
     IF (VR_V_itm ≠ NULL) THEN
          RETURN (VR_V_itm, 1) /* return the rating, along with an indication that it was computed on the
               basis of V's direct NNs (number of recursive steps = 1) */
     ENDIF
/* The VR could not be computed using the direct neighbourhood of V. Expand the search to indirect
neighbourhoods, i.e. the near neighbourhoods of the NNs. */

     FOREACH W ∈ $NN_V$
          $VR_{itm}$[W] = computeVirtualRating(W, itm)
     END /* FOREACH */
     IF (∃ X ∈ $NN_V$ : $VR_{itm}$[X] ≠ NULL) THEN
          /* Find the minimum number of recursive steps taken to compute the VRs of V's NNs; only
               VRs computed with the minimum number of steps will be considered. */
          minRadius = $\min_W$(radius($VR_{it}$[W]))
          FOREACH W ∈ $NN_V$
               IF (($VR_{itm}$[W] ≠ NULL) ∧ (radius($VR_{itm}$[W]) == minRadius)) THEN
                    VRnumerator += sim(V, W) * (value($VR_{it}$[W]) $- \overline{r_W}$)
                    VRdenominator += sim(V, W)
               ENDIF
          END /* FOREACH */
          VR_V_itm = $\overline{r_V}$ + (VRnumerator / VRdenominator)
          RETURN (VR_V_itm, 1 + minRadius)
     ELSE
          /* VR cannot be computed, return NULL */
          RETURN NULL
     ENDIF
END /* FUNCTION */
```

Listing 1. Virtual rating computation procedure

The $\mathrm{CF}_{\mathrm{VNN}}$ algorithm [3] and the $\mathrm{CF}_{\mathrm{DR}}$ algorithm [4] are recently published state-of-the-art algorithms (2019 and 2020, respectively) which also (a) target to the increase of CF prediction coverage, (b) utilize only the ratings database, requiring no additional data such as textual user reviews or user-to-user relationships retrieved from social networks and (c) attain substantial improvements regarding coverage while improving—to a small extent—rating prediction accuracy [3, 4].

For the quantification and comparison of rating prediction accuracy, two widely used metrics have been employed, namely the mean absolute error (MAE) and the root mean squared error (RMSE) [1, 2]. Obtaining and analysing both metrics provide more comprehensive insight on the rating prediction accuracy achieved by each parameter setting, as the MAE metric penalizes both small and large errors at a uniform scale, whereas the RMSE metric punishes errors of greater magnitude more strictly.

To compute the MAE and RMSE metrics, we applied the widely used "hide one" method [1, 2]: iteratively, one item in the database was concealed and then a prediction for this rating was computed on the basis of the non-hidden ratings. This process was performed for all ratings in the database. We also conducted a second experiment where, for each user, only her most recent rating was concealed, and the value of the hidden rating was predicted on basis of the non-hidden ratings. The comparison between the results of the two experiments showed that the magnitude of the differences for both the MAE and the RMSE metric were less than 2.8% in all cases, and henceforth we present only the results of the first experiment for brevity. Prediction coverage was calculated as the ratio of the users for whom personalized predictions could be computed to the total number of users.

All experiments were run on eight datasets. Seven of these datasets have been sourced from Amazon [40, 41], while the eighth dataset has been retrieved from MovieLens [42, 43]. These are the exact same datasets the $\mathrm{CF}_{\mathrm{VNN}}$ algorithm was tested in [3], however, in this paper, we utilized the respective 5-core counterpart datasets (n.b.: in a 5-core dataset, each user and item included in the dataset has at least 5 reviews). Within the paper introducing the $\mathrm{CF}_{\mathrm{VNN}}$ algorithm [3], experiments were executed against the initial datasets, after dropping users having less than 10 ratings each. However, the 5-core dataset was deemed more appropriate, since in a considerable number of cases (ranging from 5 to 15% in the Amazon-sourced datasets) only a single rating existed in the dataset for the respective item, hence when this rating was concealed, rating prediction computation for the specific item was not possible.

The eight datasets utilized in our experimental evaluation are presented in Table 2. They exhibit the following properties:

1. They are broadly used for experimentation and benchmarking in CF research,
2. They are up to date (published between 1996 and 2016) and
3. They cover a wide range of item domains (e.g. office supplies, food, movies and music), while their size varies from 1.4 to 216 MB in plain text format.

The statistics about the number of users, number of items and number of ratings relate to the size of the dataset; these are included to demonstrate the applicability of the algorithm to datasets of different sizes. The density statistic denotes percentage of entries in the user-rating matrix that are non-empty, i.e. the percentage of (user, item) pairs where the user has entered a rating for the item. Density is directly correlated to sparsity, since sparsity = 1 − density, and the low rating prediction coverage targeted by the proposed algorithm is mainly exhibited in low density datasets. Finally, the average #ratings/user statistic is given since it has been shown to affect the performance of CF algorithms when sparsity-related effects are considered [24, 44]. The proposed algorithm does not necessitate or utilize any additional dataset features, aiming to be applicable to all CF datasets, hence the statistics shown in Table 2 are confined to the ones pertinent to the algorithm, as discussed above.

Table 2 Dataset summary

Dataset name	#Users	#Items	#Ratings	Avg. #ratings/user	Density (%)	DB size (in text format) (MB)
Amazon "Videogames" [40]	24 K	11 K	232 K	9.7	0.089	5
Amazon "CDs and Vinyl" [40]	75 K	64 K	1.1 M	14.7	0.023	25
Amazon "Movies and TV" [40]	124 K	50 K	1.7 M	13.7	0.027	40
Amazon "Books" [40]	604 K	368 K	8.9 M	14.7	0.004	216
Amazon "Digital Music" [40]	5.5 K	3.5 K	65 K	11.8	0.327	1.4
Amazon "Office Supplies" [40]	5 K	2.5 K	53 K	10.6	0.448	1.1
Amazon "Grocery and Gourmet Food" [40]	15 K	9 K	151 K	10.1	0.118	3.4
MovieLens "Latest 100 K—recommended for education and development" [42]	670	9 K	100 K	166	1.85	2.2

Finally, we compare the CF_{VR} algorithm with the algorithm presented in [39], in terms of rating prediction accuracy. The algorithm presented in [39], is also a recently published (2019) state-of-the-art algorithm that targets rating prediction accuracy increase, by handling noise in the rating data that distort the CF systems' accuracy, a problem extremely frequent to sparse CF dataset and has found to be more efficient than other algorithms of the same category.

For our experiments, we used a laptop with the following characteristics: an Intel N5000 CPU operating at a frequency of 1.1 GHz; 8 GB of DDR3 RAM; and a 256 GB solid-state drive with a transfer rate equal to 560MBps. This laptop both stored the eight datasets and executed the rating prediction algorithms. The process for conducting all experiments was as follows:

1. Initially, the relevant dataset was loaded from the text-formatted file into dynamic hash indexes. Using hash structures for indexing data within the main memory provided low lookup times.
2. The similarities between users were computed and the NNs for each user were discerned.
3. The computeVirtualRating algorithm shown in Listing 1 was employed to compute VR ratings.
4. After the computation of VRs concluded, the computed VRs were injected into the hash structures already hosting the user–item rating database contents. Subsequently, the similarities between users were recomputed.
5. Finally, rating predictions were computed by applying the "hide one" method described above, and the coverage and error metrics were computed.

Coverage Increase

The first experiment is aimed at determining the number of recursive steps that the VR computation algorithm should take to reach a satisfactory increase in coverage (close to that obtained by applying an exhaustive search of each user's direct and indirect near neighbourhoods), while additionally maintaining computational efficiency. More specifically, we have implemented an exhaustive search algorithm (Floyd–Warshall [45]) to find the maximum increase in coverage that can be obtained through the application of the CF_{VR} algorithm. This maximum is reached when the algorithm in Listing 1 is permitted to search the complete direct and indirect near neighbourhood of each user *U* to compute VRs, which will then be exploited to formulate recommendations to *U*. To promote efficiency, however, it is possible to confine the search to a subset of each user's (direct and indirect) near neighbourhood; in this paper, we use the neighbourhood radius as a criterion for confining the search. For each dataset, we started with a NN radius equal to one, and extended the radius until the coverage obtained reached or exceeded the 99% of the maximum coverage that could be obtained by the CF_{VR} algorithm (i.e. when the VR computation algorithm searched the complete direct and indirect near neighbourhood of each user). The results obtained for each dataset in the context of this experiment are reported in the following paragraphs:

The Amazon "Videogames" Dataset

Figure 2 illustrates the rating prediction coverage increase achieved by the CF_{VR} algorithm, for both similarity metrics, using the performance of the plain CF algorithm as a yardstick, for the Amazon "Videogames" dataset. In both cases, we can observe that the maximum coverage increase achieved by the CF_{VR} algorithm (exhaustive search) is 39.49% under the PCC similarity metric and 23.22% under the CS metric. When the Rad_{NN} is set to 1 (i.e. only the direct near neighbours are considered for VR computation), the coverage increase achieved by the CF_{VR} algorithm is equal to the 81.2% of the maximum under the PCC similarity metric and 74.1% of the maximum under the CS similarity metric. When Rad_{NN} increases to 2, the coverage increase is very close to the maximum under both similarity metrics (99.87% for the PCC and 99.91% for the CS). Clearly, setting Rad_{NN} to 2 achieves a near-to-maximum coverage increase, while at the same time removing the need to exhaustively search the (indirect) near neighbourhood, considerably improving, therefore, the performance of the algorithm.

In Fig. 2, we can also observe that the CF_{VR} algorithm with $Rad_{NN}=2$, surpasses the performance of the CF_{VNN} algorithm by 9.90% under the PCC similarity metric and by 6.00% under the CS similarity metric. This is due to the fact that the CF_{VNN} algorithm presented in [3] decreases

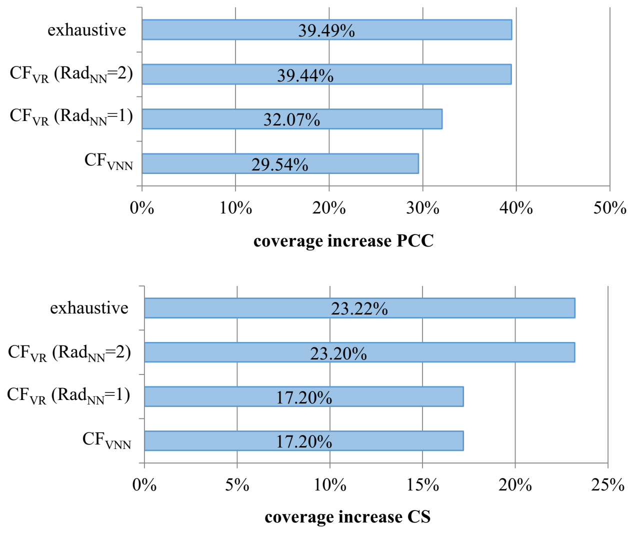

Fig. 2 Coverage increase for both similarity metrics for the Amazon "Videogames" Dataset

the sparsity of the user–item rating matrix at a granularity of users (through the introduction of virtual users), however, in some cases, these VNNs bear a negative correlation with "real" users and cannot, therefore, contribute to the formulation of predictions for these users, as shown in the example presented in the introduction section. In contrast, the CF_{VR} algorithm reduces the user–item rating matrix sparsity at a granularity of individual ratings (insertion of VRs), which are always usable for prediction formulation.

The Amazon "CDs and Vinyl" Dataset

Figure 3 illustrates the rating prediction coverage increase achieved by the CF_{VR} algorithm, for both similarity metrics, again using the performance of the plain CF algorithm as a yardstick, for the Amazon "CDs and Vinyl" dataset. For the PCC metric, the maximum coverage increase attained by the CF_{VR} algorithm is 36.41% (exhaustive setting); when Rad_{NN} is set to 1, the coverage increase achieved is 81.5% of this maximum, while a further increase of Rad_{NN} to 2 raises the coverage increase to the 99.91% of the maximum. Similarly, under the CS similarity metric, setting $Rad_{NN} = 1$ delivers a coverage increase equal to the 86.94% of the maximum (which is 23.12%), whereas setting $Rad_{NN} = 2$ enlarges the achieved coverage increase to the 99.83% of the maximum. Evidently, the setting $Rad_{NN} = 2$ warrants sufficient coverage increase in this dataset too, while reducing the computational cost by pruning the users' NN search in the VR computation phase.

In Fig. 3, we can also notice that the CF_{VR} algorithm under the setting $Rad_{NN} = 2$ exceeds the performance of the CF_{VNN} algorithm by 8.98% under the PCC similarity metric and by 2.98% under the CS similarity metric.

The Amazon "Movies and TV" Dataset

Figure 4 illustrates the rating prediction coverage increase achieved by the CF_{VR} algorithm, for both similarity metrics, again using the performance of the plain CF algorithm as a yardstick, for the Amazon "Movies and TV" dataset. In both cases, we can notice that using a setting of $Rad_{NN} = 2$ suffices to achieve a coverage increment practically identical to the one delivered when an exhaustive neighbourhood search is employed. The coverage increments reaped in this dataset are smaller than those observed in the cases of the previous two datasets: this is owing to the fact that the coverage of the plain CF algorithm was already relatively high (77% for the PCC metric and 91.2% for the CS metric), hence the improvement margins were limited.

In Fig. 4, we can observe that the CF_{VR} algorithm with $Rad_{NN} = 2$ outperforms CF_{VNN} algorithm by 4.82% under the PCC similarity metric and by 1.20% under the CS similarity metric.

The Amazon "Books" Dataset

Figure 5 illustrates the rating prediction coverage increase achieved by the CF_{VR} algorithm, for both PCC and CS similarity metrics, using the performance of the plain CF algorithm as a yardstick, for the Amazon "Books" dataset. In both cases, we can observe that when Rad_{NN} is set to 2, the coverage increases achieved by the CF_{VR}

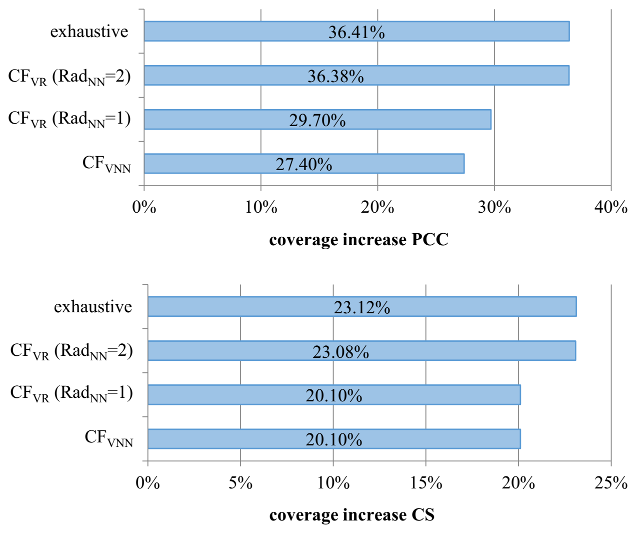

Fig. 3 Coverage increase for both similarity metrics for the Amazon "CDs and Vinyl" dataset

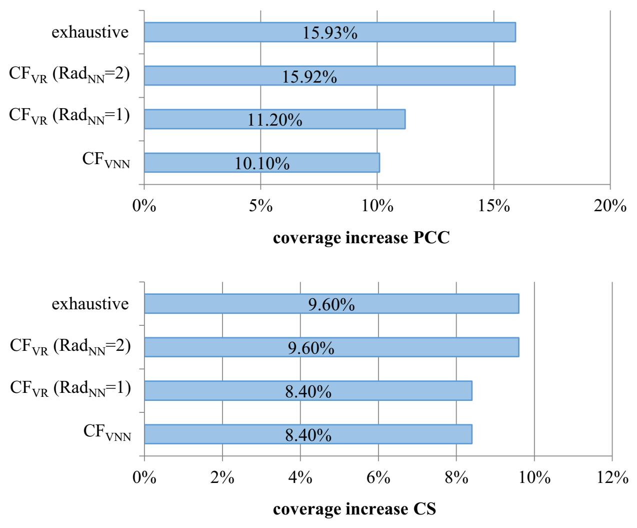

Fig. 4 Coverage increase for both similarity metrics for the Amazon "Movies and TV" dataset

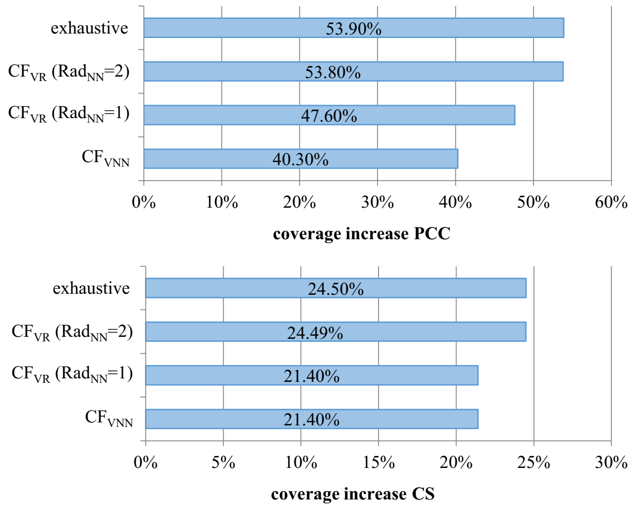

Fig. 5 Coverage increase for both similarity metrics for the Amazon “Books” dataset

algorithm is very close to the maximum ones obtained under an exhaustive NN search (99.81% of maximum coverage increase under the PCC similarity metric, and 99.96% under the CS similarity metric). The high coverage increases observed in this dataset are owing to the low density of the dataset (0.004%), which confined the coverage of the plain CF algorithm to 52% under the PCC similarity metric, introducing thus a wide improvement margin.

In Fig. 5, we can observe that the CF$_{VR}$ algorithm with Rad$_{NN}$ = 2 outperforms CF$_{VNN}$ algorithm by 13.50% under the PCC similarity metric and by 3.09% under the CS similarity metric.

The Amazon “Digital Music” Dataset

Figure 6 illustrates the rating prediction coverage increase achieved by the CF$_{VR}$ algorithm, for both PCC and CS similarity metrics, again using the performance of the plain CF algorithm as a yardstick, for the Amazon “Digital Music” dataset. In both cases, we can clearly see that setting Rad$_{NN}$ = 2 is sufficient to achieve a coverage increase of at least 99.83% of the coverage increase obtained when using an exhaustive NN search in the VR computation phase.

The coverage of the plain CF algorithm in this dataset was initially relatively high (e.g., under the CS similarity metric the plain CF algorithm exhibited a coverage of 92.9%), owing to the increased density of the dataset, hence the improvement margin was low; yet again, the CF$_{VR}$ algorithm still managed to further extend the achieved coverage.

In Fig. 6, we can observe that the CF$_{VR}$ algorithm with Rad$_{NN}$ = 2 outperforms CF$_{VNN}$ algorithm by 4.76% under the PCC similarity metric and by 0.23% under the CS similarity metric.

The Amazon “Office Supplies” Dataset

Figure 7 illustrates the rating prediction coverage increase achieved by the CF$_{VR}$ algorithm, for both PCC and CS similarity metrics, using the performance of the plain CF algorithm as a yardstick, for the Amazon “Office Supplies” dataset. In both cases, we can observe that when Rad$_{NN}$ is set to 2, the coverage increases achieved by the CF$_{VR}$ algorithm is very close to the maximum ones obtained under an exhaustive NN search (99.93% of maximum coverage increase under the PCC similarity metric, and 99.79% under the CS similarity metric).

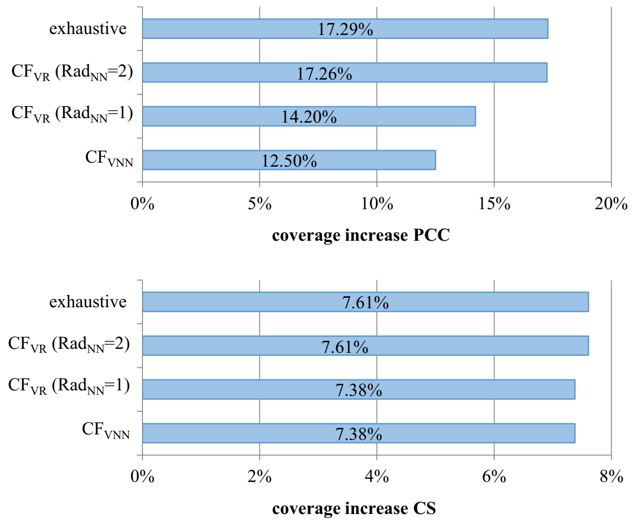

Fig. 6 Coverage increase for both similarity metrics for the Amazon “Digital Music” dataset

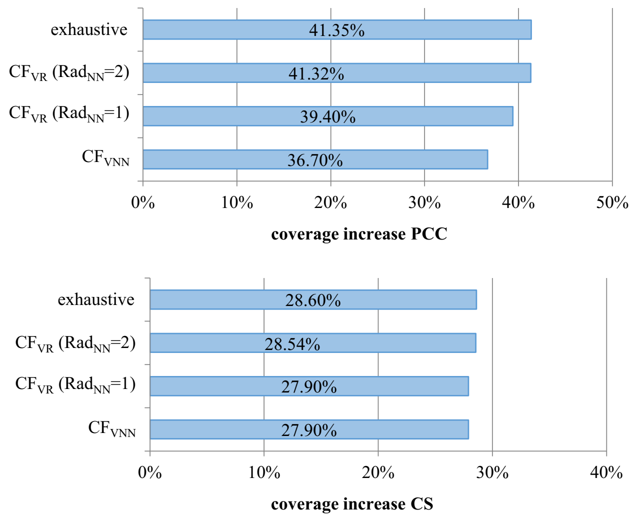

Fig. 7 Coverage increase for both similarity metrics for the Amazon “Office Supplies” dataset

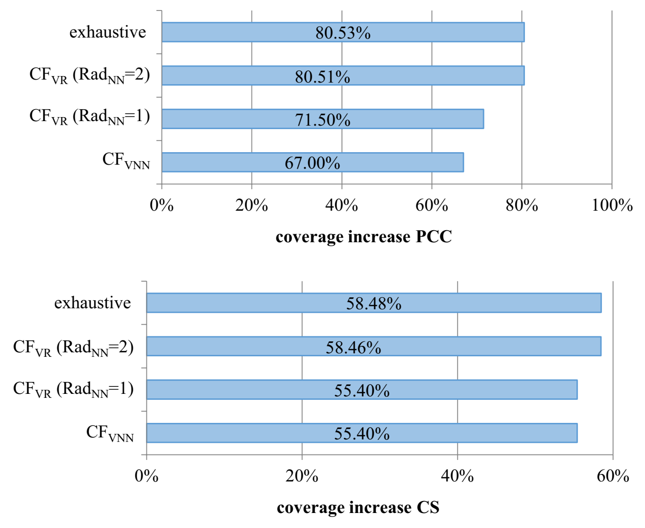

Fig. 8 Coverage increase for both similarity metrics for the Amazon "Grocery and Gourmet Food" dataset

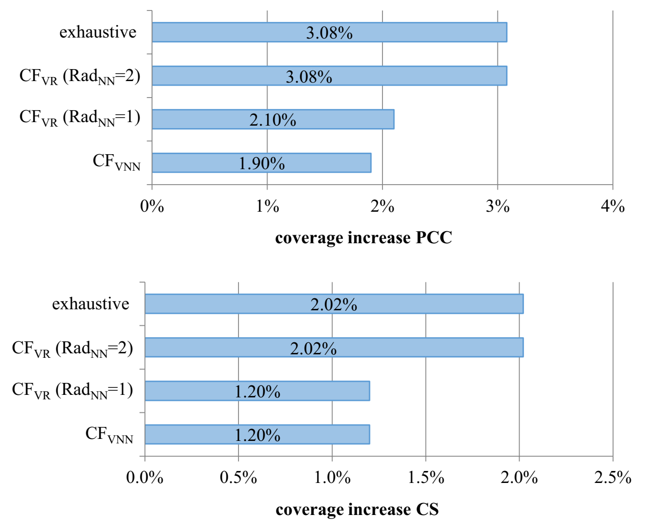

Fig. 9 Coverage increase for both similarity metrics for the MovieLens "Latest 100 K: Recommended for education and development" dataset

In Fig. 7, we can observe that the CF_{VR} algorithm with $Rad_{NN}=2$ outperforms CF_{VNN} algorithm by 4.62% under the PCC similarity metric and by 0.64% under the CS similarity metric.

The Amazon "Grocery and Gourmet Food" Dataset

Figure 8 illustrates the rating prediction coverage increase achieved by the CF_{VR} algorithm, for both PCC and CS similarity metrics, again using the performance of the plain CF algorithm as a yardstick, for the Amazon "Grocery and Gourmet Food" dataset. In both cases, we can clearly see that setting $Rad_{NN}=2$ is sufficient to achieve a coverage increase of at least 99.98% of the coverage increase obtained when using an exhaustive NN search in the VR computation phase.

It has to be noted that, under both similarity metrics, the initial plain CF coverage was relatively low (44.6% and 63.1% for the PCC and the CS, respectively). As can be seen in Fig. 8, by setting $Rad_{NN}=2$, the coverage was found to increase by 80.51% and 58.46%, for the PCC and the CS similarity metrics, respectively, reaching 80.5% and 99.99% in absolute numbers, correspondingly.

In Fig. 8, we can observe that the CF_{VR} algorithm with $Rad_{NN}=2$ outperforms CF_{VNN} algorithm by 13.51% under the PCC similarity metric and by 3.06% under the CS similarity metric.

The MovieLens "Latest 100 K: Recommended for Education and Development" Dataset

Figure 9 illustrates the rating prediction coverage increase achieved by the CF_{VR} algorithm, for both PCC and CS similarity metrics, again using the performance of the plain CF algorithm as a yardstick, for the MovieLens "Latest 100 K: Recommended for education and development" dataset. Since this dataset has very high density (avg. #Ratings-to-Users ratio = 166, more than 10 times higher than the respective ones of the Amazon datasets), the coverage achieved by the plain CF algorithm is very high (over 94% under both similarity metrics), and consequently the improvement margin is considerably limited. Nevertheless, the CF_{VR} algorithm still delivers a coverage increment, indicating that the proposed algorithm can offer coverage gains even in datasets where the initial coverage is already high.

In both cases, we can see that setting $Rad_{NN}=2$ suffices to obtain a coverage increase practically identical to that of the one achieved using an exhaustive near neighbourhood search in the VR computation phase.

In Fig. 9, we can observe that the CF_{VR} algorithm with $Rad_{NN}=2$ outperforms CF_{VNN} algorithm by 1.18% under the PCC similarity metric and by 0.82% under the CS similarity metric.

Determining the Optimal Weights for the Virtual Predictions

After having established (a) the capability of the proposed algorithm to offer coverage increments in every dataset tested, regardless of its density and the coverage delivered by the plain CF algorithm and (b) that the setting of $Rad_{NN}=2$ suffices to achieve a coverage increase practically identical to the maximum increase that can be obtained by the CF_{VR} algorithm (i.e. the one obtained under an exhaustive near neighbourhood search in the VR computation phase), we conducted a second experiment aiming at determining the optimal setting for the w_{rat} computation function f (c.f. Eq. 4).

$$f\left(\mathrm{vr}_{V,i}\right) = \operatorname*{avg}_{W \in \text{contributors}\left(\mathrm{vr}_{V,i}\right)}\left(\mathrm{sim}_{\text{mul}}(V, W)\right). \tag{6}$$

The multiplicative contributor similarity $\mathrm{sim}_{\text{mul}}(V, W)$ between a user V and a contributor of an explicitly entered rating W is computed as follows:

$$\mathrm{sim}_{\text{mul}}(V,W) = \begin{cases} \mathrm{sim}(V,W) & \text{if} \quad W \in \mathrm{NN}(V) \\ \mathrm{sim}(V,X) * \mathrm{sim}(X,W) & \text{if} \quad W \notin \mathrm{NN}(V) \wedge X \in \mathrm{NN}(V) \wedge W \in \mathrm{NN}(X) \end{cases}, \tag{7}$$

More specifically, for each setting of the w_{rat} computation function f, the rating prediction accuracy achieved by the particular setting was measured in terms of the MAE and the RMSE metrics, as described at the beginning of the "Experimental evaluation" section. In this experiment, we have set $\mathrm{Rad}_{\mathrm{NN}} = 2$. While in the experiment, we tested and analysed more than 20 weight value combinations, in the rest of this subsection we report only on the most indicative ones, for conciseness purposes.

Figure 10 illustrates the rating prediction error reduction achieved under the following settings of function f:

1. *Equivalence to explicitly entered ratings* (*Eq-Expl*) Virtual ratings are treated equally to explicitly entered ratings, by setting their weight to 1.0. Formally, $f(\mathrm{vr}_{V,i}) = 1.0$.
2. *Average of multiplicative contributor similarities* (*Avg-Mul-Contr*) The weight of a virtual rating $\mathrm{vr}_{V,i}$ is computed as the average of the multiplicative similarities between user V and each of the users that have contributed an explicitly entered rating to the computation of $\mathrm{vr}_{V,I}$, i.e.

where sim denotes the currently employed user similarity metric (PCC or CS, c.f. Eqs. (1) and (2), respectively). Effectively, the similarity between a user and her direct neighbours is expressed by the currently employed user similarity metric, while for contributors W not within the near neighbourhood of user V, the shortest path between V and W within the near neighbourhood graph is determined, and the similarity between V and W is set to the product of the weights of the edges along this path (edge weights correspond to the similarity of the users connected by the edge). Note that since $\mathrm{Rad}_{\mathrm{NN}} = 2$, the maximum length of such a path will be equal to 2.

3. *Average of maximal contributor similarities* (*Avg-Max-Contr*) Similar to the case (2), above, with the difference that for contributors W not within the near neighbourhood of user V, the similarity between V and W is set to the maximum of the weights of the edges along the shortest path between V and W. Formally, the similarity $\mathrm{sim}_{\max}(V, W)$ between a user V and a contributor of an explicitly entered rating W is expressed as

$$\mathrm{sim}_{\max}(V,W) = \begin{cases} \mathrm{sim}(V,W) & \text{if } W \in \mathrm{NN}(V) \\ \max(\mathrm{sim}(V,X),\mathrm{sim}(X,W)) & \text{if } W \notin \mathrm{NN}(V) \wedge X \in \mathrm{NN}(V) \wedge W \in \mathrm{NN}(X) \end{cases}. \tag{8}$$

4. *Average of minimal contributor similarities* (*Avg-Min-Contr*) Similar to the case (2), above, with the difference that for contributors W not within the near neighbourhood of user V, the similarity between V and W is set to the minimum of the weights of the edges along the shortest path between V and W. Formally, the similarity $\mathrm{sim}_{\min}(V, W)$ between a user V and a contributor of an explicitly entered rating W is expressed as

$$\mathrm{sim}_{\min}(V,W) = \begin{cases} \mathrm{sim}(V,W) & \text{if } W \in \mathrm{NN}(V) \\ \min(\mathrm{sim}(V,X),\mathrm{sim}(X,W)) & \text{if } W \notin \mathrm{NN}(V) \wedge X \in \mathrm{NN}(V) \wedge W \in \mathrm{NN}(X) \end{cases}. \tag{9}$$

5. *Average of mean contributor similarities* (*Avg-Mean-Contr*) Similar to the case (2), above, with the difference that for contributors W not within the near neighbourhood of user V, the similarity between V and W is

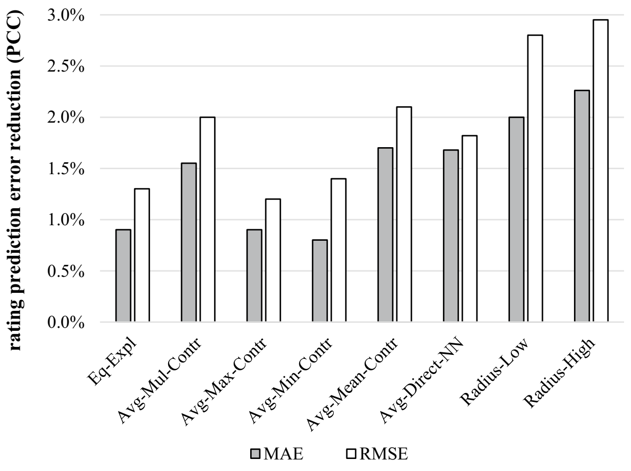

Fig. 10 Prediction error reduction under different VR weight parameter computation settings, using the PCC similarity metric

set to the mean of the weights of the edges along the shortest path between V and W. Formally, the similarity $\text{sim}_{\text{mean}}(V, W)$ between a user V and a contributor of an explicitly entered rating W is expressed as

$$\text{sim}_{\text{mean}}(V, W) = \begin{cases} \text{sim}(V, W) & \text{if} \quad W \in \text{NN}(V) \\ \frac{\text{sim}(V,X)+\text{sim}(X,W)}{2} & \text{if} \quad W \notin \text{NN}(V) \wedge X \in \text{NN}(V) \wedge W \in \text{NN}(X) \end{cases}. \tag{10}$$

6. *Average of direct neighbour similarities* (*Avg-Direct-NN*) The weight of a virtual rating $\text{vr}_{V,i}$ is set to the average of the similarities of user V to her direct neighbours involved in the computation of $\text{vr}_{V,i}$. A direct neighbour W of V is considered to be involved in the computation of $\text{vr}_{V,i}$ if either (a) W has contributed an explicitly entered rating to the computation of $\text{vr}_{V,i}$ or (b) user X is a near neighbour of W and X has contributed an explicitly entered rating to the computation of $\text{vr}_{V,i}$. The set of all direct neighbours of V that are involved in the computation of $\text{vr}_{V,i}$ will be denoted as $\text{IDN}(\text{vr}_{V,i})$. Formally, the weight $f(\text{vr}_{V,i})$ is computed as:

$$f\left(\text{vr}_{V,i}\right) = \underset{W \in \text{IDN}(\text{vr}_{V,i})}{\text{avg}} (\text{sim}(V, W)). \tag{11}$$

7. *Radius-dependent, low weight* (*Radius-Low*) The weight of a virtual rating is solely dependent on the radius of the neighbourhood explored to locate explicitly entered rating contributors as follows:

$$f(\text{vr}) = \begin{cases} 0.5 & \text{if radius(vr)} = 1 \\ 0.25 & \text{if radius(vr)} = 2 \end{cases}. \tag{12}$$

Larger values of radius (vr) are not considered, since in this experiment, we have set $\text{Rad}_{\text{NN}} = 2$.

8. *Radius-Dependent, High Weight* (*Radius-High*) Similar to case 6, above, however, the values assigned to f(vr) are higher:

$$f(\text{vr}) = \begin{cases} 0.75 & \text{if radius(vr)} = 1 \\ 0.5 & \text{if radius(vr)} = 2 \end{cases}. \tag{13}$$

Recall that in all cases, the weight of real ratings is always equal to 1.0 (c.f. Eq. 4).

Figure 10 depicts the average prediction errors reductions across all datasets, under both MAE and RMSE quantification metrics and when employing the PCC similarity metric, for the settings of function f listed above. In Fig. 10, we can observe that the setting for function f that delivers the highest prediction errors reductions, is the one denoted as Radius-High, i.e. the setting where the weight of a VR depends only on the radius of the neighbourhood exploited for its computation and assigning a weight equal to 0.75 to VRs that have been computed on the basis of the direct near neighbourhood (radius(vr) = 1) and a weight equal to 0.5 to VRs that have been computed on the basis of a near neighbourhood of a radius equal to 2. Under the aforementioned settings, an average MAE reduction of 2.26% and an average RMSE reduction of 2.95% are harvested. Notably, the results were relatively consistent across all datasets, in the sense that the ranking of the tested settings of the f function was almost the same for all the datasets tested. The results obtained for each individual dataset, concerning the optimal setting identified in this experiment, are shown and discussed in more detail in the next subsection, where the proposed algorithm is compared with the CF_{VNN} algorithm presented in [3] and the CF_{DR} algorithm presented in [4]. The runner-up is the one denoted as radius-low, i.e. the counterpart of Radius-High that assigns lower weights to VRs [equal to 0.5 and 0.25 when radius(vr) = 1 and radius(vr) = 2, respectively]. Considering the other settings, the mean and multiplicative contributor averages (Avg-Mean-Contr and Avg-Mul-Contr) are ranked in the third and fourth place, respectively.

When the CS similarity metric is employed, the same setting of function f Radius-High has been again found to achieve the highest improvements regarding rating prediction accuracy, for both the MAE and the RMSE metrics. Using this setup, the error metric reductions are 2.95% for the MAE metric and 2.09%. Again, Radius-High is followed by Radius-Low, Avg-Mean-Contr and Avg-Mul-Contr, in that order.

Execution Time Analysis

In this section, we analyze the execution time of the proposed algorithm, focusing on the overhead introduced due to the execution of two additional computation steps in comparison to the plain CF algorithm, namely (a) the computation of virtual ratings and (b) the re-computation of user similarities, considering the VR-enhanced user–item rating matrix.

The overall overhead from steps (a) and (b) listed above ranges from 5 to 40%, depending on the dataset and the similarity metric used. The smallest overheads were observed for the "Amazon videogames" and "Amazon grocery" datasets, and the largest one for the Movielens "Latest 100 K: Recommended for education and development" dataset; in general, the overhead was found to increase along with the statistical measure "Avg. #ratings/user" of the dataset. This is attributed to the fact that more real ratings lead to greater numbers of NNs, hence neighbourhood search for the computation of VRs requires more time. Furthermore, greater numbers of real ratings and NNs implies the computation of greater numbers of VRs, and this in turn increases the time needed for the re-computation of user-to-user similarities, since the re-computation of these similarities operates on more extensive data (the union of the real and virtual ratings).

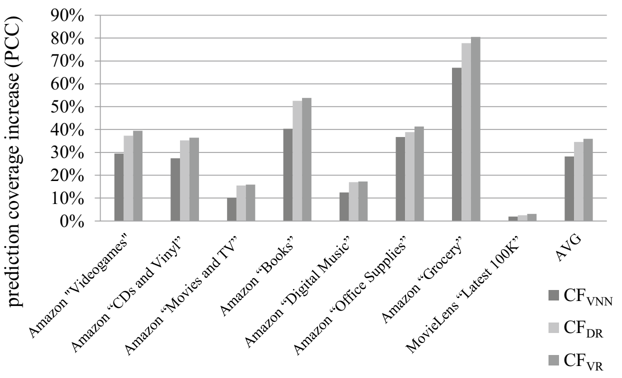

Fig. 11 Coverage increase for the different datasets, under the PCC user similarity metric

two users to be considered in the similarity computation of step (b) above are $(I(U) \cup I_V(U)) \cap (I(V) \cup I_V(U))$. This set can be subdivided to two disjoint subsets:

- The items for which both users have entered a real rating, which are given by the expression $\mathrm{CR}(U, V) = I(U) \cap I(V)$ and
- The items where at least one of the ratings is a virtual one, which are given by the expression $\mathrm{CV}(U, V) = ((I(U) \cup I_V(U)) \cap (I(V) \cup I_V(U)) - (I(U) \cap I(V)))$

Considering these subsets, Eq. (2) can be rewritten as:

$$\mathrm{sim}_{\mathrm{CS}}(U,V) = \frac{\sum_{k \in \mathrm{CR}(U,V)} r_{U,k} * r_{V,k} + \sum_{k \in \mathrm{CV}(U,V)} r_{U,k} * r_{V,k}}{\sqrt{\sum_{k \in \mathrm{CR}(U,V)} \left(r_{U,k}\right)^2 + \sum_{k \in \mathrm{CV}(U,V)} \left(r_{U,k}\right)^2} * \sqrt{\sum_{k \in \mathrm{CR}(U,V)} \left(r_{V,k}\right)^2 + \sum_{k \in \mathrm{CV}(U,V)} \left(r_{V,k}\right)^2}}. \tag{14}$$

It is worth noting here that the steps (a) and (b) above are typically performed in an offline fashion [31], thus execution time is not critical. The online part of the algorithm, which is effectively limited to the rating prediction formulation phase, based on the VR-enhanced user–item rating matrix, taking into account the similarities computed in step (b), is only slightly penalized, by a factor ranging from 0.4 to 1.3%, depending on the number of virtual ratings computed by step (a) above. Using a commodity laptop with the specifications listed in the "Experimental evaluation" section, a rating prediction is formulated in less than 10 ms, while additionally the rating prediction computation task is directly parallelizable, with concurrent rating prediction computation requests being assigned to different execution cores.

The similarity metric employed affects the re-computation of the similarities (step b, above) as follows: consider users U and V, where $I(U)$ and $I(V)$ denote the items that each user has rated, and $I_V(U)$ and $I_V(V)$ denote the items for which virtual ratings have been computed for each of the users, using the algorithm presented in Listing 1. Then, the items commonly rated by the

However, the first term of the sum in the nominator, as well as the first term in each of the terms in the denominator correspond to the respective quantities used for the computation of the original similarity between U and V, during the preparatory step for the computation of virtual ratings. Hence, these partial results can be cached from that phase and reused for the computation of user similarity considering the VR-enhanced user–item rating matrix, reducing the time needed for this step by approximately 55% on average.

When the Pearson coefficient similarity metric is used, according to Eq. (1), the mean of each user's ratings is subtracted from the ratings of the corresponding user, and the difference is used in the similarity computation formula. At this point, two options are available:

1. The mean of real user ratings is used; under this option, the optimization discussed above can be directly used;

2. The mean of both real and virtual ratings is used; under this option, the optimization presented above is not applicable.

According to experiments conducted, option (2) above exhibits slightly superior performance compared to option (1), with the MAE improvement of option (2) being 0.2% larger than the MAE improvement of option (1) on average, in absolute figures. Therefore, a trade-off between performance and accuracy improvement exists; considering, however, that similarity computation is performed in an offline fashion, option (2) is preferable.

Comparison with Previous Work

After having determined the optimal parameter values for the operation of the CF_{VR} algorithm (i.e. the optimal radius of the near neighbourhood to explored for the computation of VRs, as well as the virtual rating weight computation function that delivers the highest prediction error reduction), we elaborate on the algorithm performance evaluation results, considering rating prediction coverage and accuracy, using the eight datasets summarized in Table 2. For both aspects, the plain CF algorithm is used as a performance baseline. Besides reporting on the improvements coverage and accuracy attained by the CF_{VR} algorithm introduced in this paper, we comparatively assess its performance against the performance of the CF_{VNN} algorithm introduced in [3] and the CF_{DR} algorithm introduced in [4]. Both CF_{VNN} and CF_{DR} (a) are state-of-the-art algorithms focusing on addressing data sparsity issues and increasing rating prediction coverage, (b) accomplish substantial improvements regarding coverage, while improving—to a small extent—rating prediction accuracy (c) do not require any additional data (e.g. such as textual user reviews or user-to-user relationships retrieved from social networks).

Figure 11 illustrates the results obtained regarding the prediction coverage increase, when user-to-user similarity is quantified using the PCC metric. We can notice that the CF_{VR} algorithm, presented in this paper, achieves an average prediction coverage increase equal to 36%, exceeding the performance of the CF_{VNN} algorithm [3] by 7.8% and that of the CF_{DR} algorithm by 1.36%; the relative improvement against the CF_{VNN} algorithm, computed as $\frac{\text{Improvement}(CF_{VR})-\text{Improvement}(CF_{VNN})}{\text{Improvement}(CF_{VNN})}$, is equal to 27.7%, while the relative improvement against the CF_{DR} algorithm is equal to 3.9%.

Similarly, Fig. 12 presents the respective results obtained regarding prediction coverage increase, when user-to-user similarity is quantified using the CS metric. In this case, we can notice that the CF_{VR} algorithm, presented in this paper, achieves an average prediction coverage increase equal to 22.2%, exceeding the performance of the CF_{VNN} algorithm presented in [3] by 2.3% and the performance of the CF_{DR} algorithm [4] by

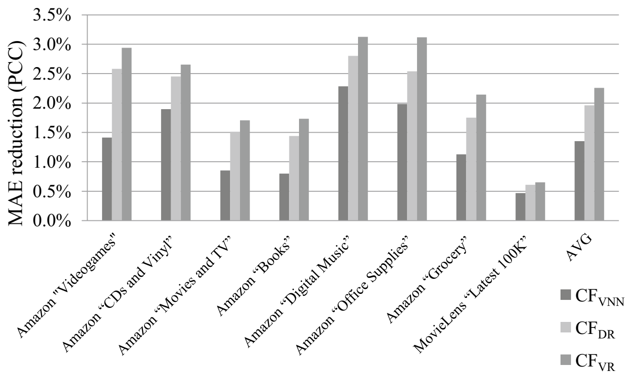

Fig. 13 MAE reduction for the different datasets, under the PCC user similarity metric

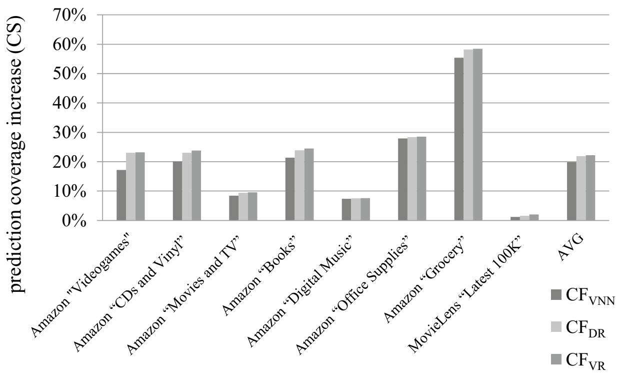

Fig. 12 Coverage increase for the different datasets, under the CS user similarity metric

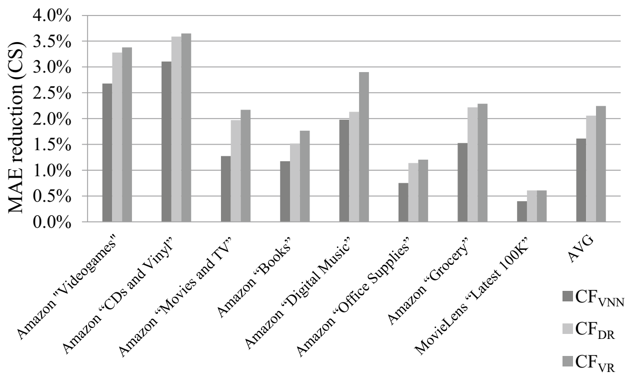

Fig. 14 MAE reduction for the different datasets, under the CS user similarity metric

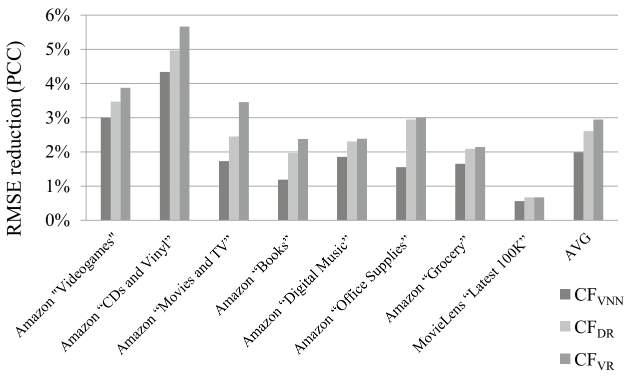

Fig. 15 RMSE reduction for the different datasets, under the PCC user similarity metric

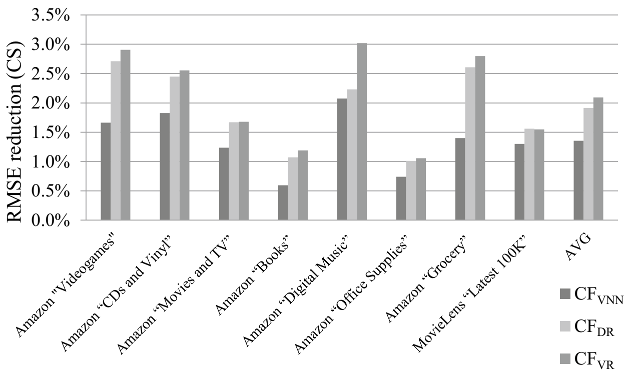

Fig. 16 RMSE reduction for the different datasets, under the CS user similarity metric

0.32% (relative improvements are equal to 11.6% and 1.47%, correspondingly).

To validate the significance of the coverage increase results, we conducted a statistical significance testing, across all datasets, between the CF_{VR} algorithm, presented in this paper, the CF_{VNN} algorithm and the CF_{DR}. The results of this experiment indicate that the proposed algorithm is shown to be statistically significant with a confidence interval of 95% under both similarity metrics. In more detail:

- Under the PCC similarity metric, $p(CF_{VR}, CF_{VNN}) = 0.008$ and $p(CF_{VR}, CF_{DR}) = 0.014$; after applying the Holm–Šídák post hoc test for p value correction for multiple tests, the adjusted p values are both equal to 0.016 and the corrected alpha for the Holm–Šídák method is equal to 0.0253; hence, a statistical significance regarding the observed differences in the performance of the CF_{VR} algorithm against the performance of both the CF_{VNN} and CF_{DR} algorithms is established.
- Under the CS similarity metric, $p(CF_{VR}, CF_{VNN}) = 0.009$ and $p(CF_{VR}, CF_{DR}) = 0.017$; after applying the Holm–Šídák post hoc test for p value correction for multiple tests, the adjusted p values are both equal to 0.0179 and the corrected alpha for the Holm–Šídák method is equal to 0.0253; therefore, a statistical significance regarding the observed differences in the performance of the CF_{VR} algorithm against the performance of both the CF_{VNN} and CF_{DR} algorithms is established.

Figure 13 illustrates the results obtained regarding the rating prediction MAE reduction, when user-to-user similarity is quantified using the PCC metric. We can notice that the average prediction MAE reduction achieved by the CF_{VR} algorithm, presented in this paper, equals to 2.26%, surpassing the performance of the CF_{VNN} algorithm by 0.91% and the performance of the CF_{DR} algorithm by 0.30% (relative improvements are equal to 67.4% and 15.2%, respectively).

Similarly, Fig. 14 illustrates the respective results obtained regarding the rating prediction MAE reduction, when user-to-user similarity is quantified using the CS metric. We can notice that while the CF_{VNN} and the CF_{DR} algorithms achieve an average MAE reduction equal to 1.61% and 2.06%, respectively, the CF_{VR} algorithm, presented in this paper, achieves an average MAE reduction equal to 2.25% (i.e. 39.8% relative improvement against CF_{VNN} and 9.2% relative improvement over CF_{DR}; the respective improvements in absolute figures are equal to 0.64% and 0.29%).

Figure 15 illustrates the measurements obtained regarding the rating prediction RMSE reduction, when user-to-user similarity is quantified using the PCC metric. We can notice that while the CF_{VNN} and CF_{DR} algorithms achieve an average RMSE reduction equal to 1.99% and 2.61, the CF_{VR} algorithm, presented in

Table 3 Statistical significance tests for accuracy

Similarity measure	MAE				RMSE			
	$p(CF_{VR}, CF_{VNN})$	$p(CF_{VR}, CF_{DR})$	Corrected p values (Holm–Šídák)	Corrected alpha (Holm–Šídák)	$p(CF_{VR}, CF_{VNN})$	$p(CF_{VR}, CF_{DR})$	Corrected p value (Holm–Šídák)	Corrected alpha (Holm–Šídák)
PCC	0.008	0.015	0.016	0.0253	0.007	0.013	0.014	0.0253
			0.016				0.014	
CS	0.005	0.012	0.001	0.0253	0.006	0.015	0.012	0.0253
			0.012				0.015	

this paper, achieves an average RMSE reduction equal to 2.95% (i.e. to 47.5% relative improvement against CF_{VNN} and 13.02% relative improvement over CF_{DR}; the respective improvements in absolute figures are equal to 0.96% and 0.34%).

Finally, Fig. 16 illustrates the measurements obtained regarding the rating prediction RMSE reduction, when user-to-user similarity is quantified using the CS metric. We can notice that while the CF_{VNN} and CF_{DR} algorithms achieve an average RMSE reduction equal to 1.35% and 1.91%, respectively, the CF_{VR} algorithm, presented in this paper, achieves an average RMSE reduction equal to 2.09% (i.e. to 54.8% relative improvement against CF_{VNN} and 9.4% relative improvement over CF_{DR}; the respective improvements in absolute figures are equal to 0.74% and 0.18%).

To establish the statistical significance of the MAE and RMSE accuracy improvement measurements, we conducted a statistical significance testing between the CF_{VR} algorithm, presented in this paper and the CF_{VNN} and CF_{DR} algorithms, considering the results obtained for all datasets. The Holm–Šídák post hoc test was applied to cater for *p* value correction for multiple tests. The results of this experiment indicate that the CF_{VR} algorithm improvements are statistically significant at a confidence level equal to 95% under both similarity metrics. Table 3 illustrates the results of the statistical tests for the MAE and RMSE metrics, under both similarity measures.

Besides the comparison of the proposed algorithm against the algorithms proposed in [3, 4], we also compare the presented algorithm with the one presented in [36], which targets noisy environment and has been found to be more efficient than other algorithms of the same category [36–39]. According to the results presented in [36], the algorithm proposed therein achieves an average improvement in MAE equal to 1.12% (ranging from 0.42 to 1.83%) against the plain CF algorithm, while it does not have any effect on the recommendation coverage. The algorithm proposed in this paper achieves an average MAE reduction equal to 2.26% combined with a coverage increase of 36% under the PCC metric, while the respective improvements under the CS metric are equal to 2.25% and 22.2%. [39] also reports on an algorithm that improves the MAE, without, however, affecting coverage. Moreover, these improvements are reported for users having 20–100 near neighbours, a condition that is not met in sparse datasets.

Conclusions and Future Work

In this paper, we presented the CF_{VR} algorithm, which is a novel CF algorithm for improving prediction coverage in sparse datasets.

The novelty behind the proposed algorithm is the introduction of the virtual ratings, which are formulated for each NN of a user *U* who cannot contribute to the prediction formulated for *U* with a real rating that effectively reduces the user–item rating matrix sparsity, thus alleviating the "grey sheep" problem, all sparse CF datasets suffer from. The procedure for virtual rating creation considers the direct neighbourhood of the user, as well as his/her indirect neighbourhood, while each VR is assigned a weight, based on the aspects of the neighbourhood that has contributed to its formulation and reflects the degree of confidence to the value of the VR. The incorporation of the weight has been shown to increase rating prediction accuracy.

The presented algorithm has been experimentally verified using eight datasets and the evaluation results have shown that the introduction of VRs may increase prediction coverage by a factor ranging from 7.6 to 80.5% under the PCC similarity metric for sparse datasets (the actual improvement is dataset-dependent) when the VRs are computed considering the full transitive closure of the NN relationship among users. However, experiments have shown that limiting the near neighbourhood range to a value of 2 is sufficient to reach prediction coverage of at least equal to 99.91% of the maximum (i.e. the one obtained when considering the full transitive closure of the NN relationship), allowing thus to harvest significant gains in computational efficiency with a negligible effect on the prediction coverage. In parallel, the CF_{VR} algorithm achieves considerable improvements in terms of rating prediction accuracy, decreasing the MAE by 2.3% and the RMSE by 2.5% on average.

We have also comparatively evaluated performance the CF_{VR} algorithm against the CF_{VNN} [3] and the CF_{DR} algorithm [4] in terms of performance; CF_{VNN} and CF_{DR} are state-of-the-art algorithms, aiming to increase prediction coverage in sparse datasets, utilizing solely the user–item ratings database. The CF_{VR} algorithm, presented in this paper, has been shown to surpass the performance of the CF_{VNN} and the CF_{DR} algorithms, both in terms of rating prediction coverage increase and rating prediction accuracy improvement, for both the similarity metrics tested.

The proposed algorithm exhibits the practical advantages that (a) achieves to increase both coverage and accuracy, and (b) operates only using the user–item rating matrix, without requiring any additional data and thus being applicable in all cases where this elementary information is available. An identified limitation of this work is that in very sparse datasets coverage increments may be obtained, however, in some cases, the increased coverage may not exceed 80%. In these cases, the user–item rating matrix data should be supplemented with additional data that can be exploited, such as user-to-user relationships, textual reviews and so forth.

Regarding our future work, we plan to explore alternative algorithms for increasing rating prediction coverage and/or reducing rating prediction error in sparse CF datasets. Furthermore, we plan to examine these algorithms using more similarity metrics, such as the Spearman coefficient, the Euclidian distance and the Manhattan distance [46]. Finally, we will examine the extension of the CF_{VR} algorithm to accommodate and exploit additional data sources, such as social network-sourced information, textual reviews or IoT data, in order to further improve rating prediction coverage and rating prediction accuracy. In this context, existing algorithms in these areas [47–54] will be studied and adapted accordingly.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Research involving human participants and/or animals This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent This research is solely based on anonymized datasets, where no user identifiable information is present. The datasets were prepared and published in the public domain by third parties.

References

1. Balabanovic M, Shoham Y. Fab: content-based, collaborative recommendation. Commun ACM. 1997;40(3):66–72. https://doi.org/10.1145/245108.245124.
2. Ekstrand M, Riedl R, Konstan J. Collaborative filtering recommender systems. Found Trends Hum Comput Interact. 2011;4(2):81–173. https://doi.org/10.1561/1100000009.
3. Margaris D, Vasilopoulos D, Vassilakis C, Spiliotopoulos D. Improving collaborative filtering's rating prediction coverage in sparse datasets through the introduction of virtual near neighbors. In: Proceedings of the 2019 10th international conference on information, intelligence, systems and applications (IISA). 2019. p. 1–8. https://doi.org/10.1109/IISA.2019.8900678.
4. Margaris D, Spiliotopoulos D, Karagiorgos G, Vassilakis C. An algorithm for density enrichment of sparse collaborative filtering datasets using robust predictions as derived ratings. Algorithms. 2020;13(7):174. https://doi.org/10.3390/a13070174.
5. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. IEEE Comput. 2009;42(8):42–9. https://doi.org/10.1109/MC.2009.263.
6. McAuley JJ, Targett C, Shi Q, Van den Hengel A. Image-Based recommendations on styles and substitutes. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. 2015. p. 43–52. https://doi.org/10.1145/2766462.2767755..
7. Margaris D, Vassilakis C. Improving collaborative filtering's rating prediction quality in dense datasets, by pruning old ratings. In: Proceedings of the 2017 IEEE symposium on computers and communications (ISCC). 2017. p. 1168–1174. https://doi.org/10.1109/ISCC.2017.8024683..
8. Margaris D, Vassilakis C. Enhancing user rating database consistency through pruning. Trans Large Scale Data Knowl Cent Syst. 2017;XXXIV:33–64. https://doi.org/10.1007/978-3-662-55947-5_3.
9. Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A. A survey on concept drift adaptation. ACM Comput Surv. 2013. https://doi.org/10.1145/2523813.
10. Lu J, Liu A, Song Y, Zhang G. Data-driven decision support under concept drift in streamed big data. Complex Intell Syst. 2020;6(1):157–63. https://doi.org/10.1007/s40747-019-00124-4.
11. Paudel R, Eberle W. An approach for concept drift detection in a graph stream using discriminative subgraphs. ACM Trans Knowl Discov Data. 2020;14(6):1–25. https://doi.org/10.1145/3406243.
12. Gong S. A collaborative filtering recommendation algorithm based on user clustering and item clustering. J Softw. 2010;5(7):745–52.
13. Chen J, Zhao C, Uliji CL. Collaborative filtering recommendation algorithm based on user correlation and evolutionary clustering. Complex Intell Syst. 2020;6(1):147–56. https://doi.org/10.1007/s40747-019-00123-5.
14. Pham M, Cao Y, Klamma R, Jarke M. A clustering approach for collaborative filtering recommendation using social network analysis. J Univ Comput Sci. 2011;17(4):583–604. https://doi.org/10.3217/jucs-017-04-0583.
15. Kalaï A, Zayani CA, Amous I, Abdelghani W, Sèdes F. Social collaborative service recommendation approach based on user's trust and domain-specific expertise. Futur Gener Comput Syst. 2018;80:355–67. https://doi.org/10.1016/j.future.2017.05.036.
16. Margaris D, Spiliotopoulos D, Vassilakis C. Social relations versus near neighbours: reliable recommenders in limited information social network collaborative filtering for online advertising. In: Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2019). 2019. p. 1160–1167. https://doi.org/10.1145/3341161.3345620..
17. Sun J, Ying R, Jiang Y, He J, Ding Z. Leveraging friend and group information to improve social recommender system. Electron Commer Res. 2020;20(1):147–72. https://doi.org/10.1007/s10660-019-09390-3.
18. Vozalis M, Markos A, Margaritis K. A hybrid approach for improving prediction coverage of collaborative filtering. Artif Intell Appl Innov. 2009;296:491–8. https://doi.org/10.1007/978-1-4419-0221-4_57.
19. Zhang S, Yao L, Xu X. Autosvd++: an efficient hybrid collaborative filtering model via contractive auto-encoders. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval. 2017. p. 957–960. https://doi.org/10.1145/3077136.3080689..
20. Yang X, Zhou S, Cao M. An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews. Mob Netw Appl. 2020;25:376–90. https://doi.org/10.1007/s11036-019-01246-2.
21. Walek B, Fojtik V. A hybrid recommender system for recommending relevant movies using an expert system. Expert Syst Appl. 2020;158:113452. https://doi.org/10.1016/j.eswa.2020.113452.
22. Adamopoulos P. Beyond rating prediction accuracy: on new perspectives in recommender systems. In: Proceedings of the 7th ACM conference on recommender systems (RecSys '13). 2013. p. 459–462. https://doi.org/10.1145/2507157.2508073..
23. Margaris D, Kobusińska A, Spiliotopoulos D, Vassilakis C. An adaptive social network-aware collaborative filtering algorithm for improved rating prediction accuracy. IEEE Access. 2020;8:68301–10. https://doi.org/10.1109/ACCESS.2020.2981567.
24. Margaris D, Vassilakis C. Improving collaborative filtering's rating prediction coverage in sparse datasets by exploiting user dissimilarity. In: Proceedings of the 4th IEEE international conference on big data intelligence and computing. 2018. p. 1054–1059. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00150..
25. Wang P, Huang H, Zhu J, Qi L. A trust-based prediction approach for recommendation system. In: Proceedings of the world congress on services 2018, LNCS, vol. 10975. Cham, Springer. 2018. p. 157–164. https://doi.org/10.1007/978-3-319-94472-2_12..

26. Zarei MR, Moosavi MR. A memory-based collaborative filtering recommender system using social ties. In: Proceedings of the 4th international conference on pattern recognition and image analysis (IPRIA). 2019. p. 263–267. https://doi.org/10.1109/PRIA.2019.8786023..
27. Wen H, Ding G, Liu C, Wang J. Matrix factorization meets cosine similarity: addressing sparsity problem in collaborative filtering recommender system. Proc APWeb. 2014;2014:306–17. https://doi.org/10.1007/978-3-319-11116-2_27.
28. Guan X, Li C, Guan Y. Matrix factorization with rating completion: an enhanced SVD model for collaborative filtering recommender systems. IEEE Access. 2017;5:27668–78. https://doi.org/10.1109/ACCESS.2017.2772226.
29. Poirier D, Fessant F, Tellier I. Reducing the cold-start problem in content recommendation through opinion classification. In: Proceedings of the 2010 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology. 2010. p. 204–207. https://doi.org/10.1109/WI-IAT.2010.87..
30. Moshfeghi Y, Piwowarski B, Jose JM. Handling data sparsity in collaborative filtering using emotion and semantic based features. In: Proceedings of 34th international ACM SIGIR conference. 2011. p. 625–634. https://doi.org/10.1145/2009916.2010001..
31. Margaris D, Vassilakis C, Spiliotopoulos D. Handling uncertainty in social media textual information for improving venue recommendation formulation quality in social networks. Soc Netw Anal Min. 2019;64:1–19. https://doi.org/10.1007/s13278-019-0610-x.
32. Margaris D, Vassilakis C. Improving collaborative filtering's rating prediction coverage in sparse datasets by exploiting the 'friend of a friend' concept. Int J Big Data Intell. 2020;7(1):47–57. https://doi.org/10.1504/IJBDI.2020.106178.
33. O'Mahony MP, Hurley NJ, Silvestre G. Detecting noise in recommender system databases. In: Proceedings of the 11th international conference on intelligent user interfaces. 2006. p. 109–115. https://doi.org/10.1145/1111449.1111477..
34. Chung CY, Hsu PY, Huang SH. βP: a novel approach to filter out malicious rating profiles from recommender systems. Decis Support Syst. 2013;55(1):314–25. https://doi.org/10.1016/j.dss.2013.01.020.
35. Lee JS, Zhu D. Shilling attack detection-a new approach for a trustworthy recommender system. INFORMS J Comput. 2012;24(1):117–31. https://doi.org/10.1287/ijoc.1100.0440.
36. Toledo RY, Mota YC. Correcting noisy ratings in collaborative recommender systems. Knowl Based Syst. 2015;76:96–108. https://doi.org/10.1016/j.knosys.2014.12.011.
37. Yera R, Castro J, Martínez L. A fuzzy model for managing natural noise in recommender systems. Appl Soft Comput. 2016;40:187–98. https://doi.org/10.1016/j.asoc.2015.10.060.
38. Patra BK, Launonen R, Ollikainen V, Nandib S. A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. Knowl Based Syst. 2015;82:163–77. https://doi.org/10.1016/j.knosys.2015.03.001.
39. Bag S, Kumar S, Awasthi A, Tiwaria K. A noise correction-based approach to support a recommender system in a highly sparse rating environment. Decis Support Syst. 2019;118:46–57. https://doi.org/10.1016/j.dss.2019.01.001.
40. Amazon product data. Available online: http://jmcauley.ucsd.edu/data/amazon/links.html. Accessed 4 Apr 2019.
41. McAuley JJ, Pandey R, Leskovec J. Inferring networks of substitutable and complementary products. In: Proceedings of the 21th ACM SIGKDD conference. 2015. p. 785–794. https://doi.org/10.1145/2783258.2783381.
42. MovieLens datasets. http://grouplens.org/datasets/movielens/. Accessed 4 Apr 2019.
43. Harper FM, Konstan JA. The MovieLens datasets: history and context. ACM Trans Interact Intell Syst. 2015;5(4):19. https://doi.org/10.1145/2827872.
44. Zanker M, Jessenitschnig M, Jannach D, Gordea S. Comparing recommendation strategies in a commercial context. IEEE Intell Syst. 2007;2(3):69–73. https://doi.org/10.1109/MIS.2007.49.
45. Ramadhan Z, Siahaan A, Mesran M. Prim and Floyd–Warshall comparative algorithms in shortest path problem. In: Proceedings of the joint workshop KO2PI and the 1st international conference on advance and scientific innovation. 2018. p. 47–58. https://doi.org/10.4108/eai.23-4-2018.2277598..
46. Herlocker JL, Konstan JA, Terveen LG, Riedl JT. Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst. 2004;22(1):5–53. https://doi.org/10.1145/963770.963772.
47. Tahmasebi H, Ravanmehr R, Mohamadrezaei R. Social movie recommender system based on deep autoencoder network using Twitter data. Neural Comput Appl. 2021;33:1607–23. https://doi.org/10.1007/s00521-020-05085-1.
48. Hu GN, Dai XY, Qiu FY, Xia R, Li T, Huang SJ, Chen JJ. Collaborative filtering with topic and social latent factors incorporating implicit feedback. ACM Trans Knowl Discov Data (TKDD). 2018;12(2):1–30. https://doi.org/10.1145/3127873.
49. Aivazoglou M, Roussos A, Margaris D, Vassilakis C, Ioannidis S, Polakis J, Spiliotopoulos D. A fine-grained social network recommender system. Soc Netw Anal Min. 2020;8:1–18. https://doi.org/10.1007/s13278-019-0621-7.
50. Guy I. Social recommender systems. In: Ricci F, Rokach L, Shapira B, editors. Recommender systems handbook. Boston: Springer; 2015. p. 511–43.
51. Zuo X, Liu X, Yang B. Coupled low rank approximation for collaborative filtering in social networks. IEEE Access. 2018;6:13326–35. https://doi.org/10.1109/ACCESS.2018.2806488.
52. Ojagh S, Malek MR, Saeedi S, Liang S. A location-based orientation-aware recommender system using IoT smart devices and social networks. Futur Gener Comput Syst. 2020;108:97–118. https://doi.org/10.1016/j.future.2020.02.041.
53. Subramaniyaswamy V, Manogaran G, Logesh R, Vijayakumar V, Chilamkurti N, Malathi D, Senthilselvan N. An ontology-driven personalized food recommendation in IoT-based healthcare system. J Supercomput. 2019;75(6):3184–216. https://doi.org/10.1007/s11227-018-2331-8.
54. Ren C, Chen J, Kuo Y, Wu D, Yang M. Recommender system for mobile users. Multimed Tools Appl. 2018;77(4):4133–53. https://doi.org/10.1007/s11042-017-4527-y.