Information Sciences 199 (2012) 78–92

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Using past-prediction accuracy in recommender systems

Juan F. Huete [a,*], J.M. Fernández-Luna [a], Luis M. de Campos [a], Miguel A. Rueda-Morales [b]

[a] Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática y de Telecomunicación, CITIC-UGR Universidad de Granada, C.P., 18071 Granada, Spain
[b] Oficina Transferencia Resultados de Investigación, Universidad de Granada, Granada, Spain

ARTICLE INFO

Article history:
Received 7 July 2011
Received in revised form 13 January 2012
Accepted 17 February 2012
Available online 6 March 2012

Keywords:
Recommender system
Collaborative filtering
Memory-based method
Past preference

ABSTRACT

This paper presents a new approach for memory-based collaborative filtering algorithms. In general, user-based rating prediction is a process in which each neighbor suggests a rating for the target item and the suggestions are combined by weighting the contribution of each neighbor. We present a new alternative that is independent of user rating scales and is based on what we call predictive probabilities. We explore how these probabilities can be used to select nearest neighbors for recommendations and integrate different types of dependence in the ratings. The neighborhood selection criterion depends on the capability of a user to predict past ratings. Our hypothesis is that if a user was good when predicting past ratings for an active user, then his predictions will also be helpful in recommending ratings for the same user in the future.

1. Introduction

Recommender systems (RSs) attempt to discover user preferences and learn about them to anticipate user needs. The main task is normally to offer suggestions for items that may be useful to a user, such as a book, movie, hotel or song. In recent years, interest in RSs has dramatically increased and plays an important role for internet sites such as Amazon, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, and IMDb. RSs have a promising future in many domains since personalized recommendations can be provided for users and service providers may expect an increase in sales.

Several algorithms based on different ideas and concepts have been developed to compute recommendations. These include collaborative filtering [21], which recommends an item based on the opinions of similar-minded users. To find similar users, the algorithms take as input a user $\times$ item matrix, which is usually very sparse, comprising the set of ratings given to certain observed items by each user. The ratings can be binary (0/1), representing whether a user observed or did not observe the item, or a value in a range representing the user's opinions about the item quality.

Collaborative filtering algorithms can be grouped into two main approaches [20]: memory-based algorithms, which use the entire rating matrix to make recommendations, and model-based algorithms, which use this matrix to learn (offline) an explicit model of the relationships between users (and/or items). This model is then used (online) to recommend a product to users. In this paper, we focus on memory-based algorithms, so we consider this approach in more detail. Three main steps can be considered [23]:

1. Weight all users/items with respect to a selected similarity criterion.
2. Select a subset of users/items to use as a set of predictors.

* Corresponding author.
E-mail addresses: jhg@decsai.ugr.es (J.F. Huete), jmfluna@decsai.ugr.es (J.M. Fernández-Luna), lci@decsai.ugr.es (L.M. de Campos), mruedam@ugr.es (M.A. Rueda-Morales).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.02.033

3. Compute the prediction from a weighted combination of selected neighbor ratings.

Because predictions in memory-based approaches are obtained by considering the most similar users or items, these methods are also known as neighborhood-based recommendation approaches. The way in which similarity is computed has a significant impact on the accuracy of the system. Studies and real-world implementations so far have relied on traditional vector similarity measures of the closeness between users ratings (Pearson's correlation in different formulations, cosine, Spearman's rank, etc.) [22,23,2,41,19]. The measures yielding the best results are those based on correlations: strong correlations can indicate predictive relationships, and this fact is exploited in practice by the RS. For instance, in Table 1 it is evident that Ann's and Sara's ratings are highly correlated. In particular, for a given item I_i, $\text{Ann}(r_i) = \text{Sara}(r_i) - 1$, and therefore we can use this equation to predict the rating that Ann should give to the target item I_t, denoted by $\hat{r}_t$: $\text{Ann}(\hat{r}_t) = \text{Sara}(r_t) - 1 = 3 - 1 = 2$.

However, correlation coefficients reflect only linear dependence between user rating patterns and it is possible for two sets of user ratings to be both strongly dependent and uncorrelated (correlation value equals zero). The most direct consequence is that memory-based algorithms discard these users as part of the neighborhood because they are unable to explore their predictive power. Moreover, these types of dependence are not taken into account when computing predictions using traditional memory-based approaches. To illustrate this, consider Alf's and Ann's ratings in Table 1. In this case it is evident that given that we know Alf's rating, we can predict the rating given by Ann. Thus, in this case we can guess that Ann might rate I_t as 2 because whenever Alf assigns a rating of 4, Ann uses a rating of 2. These types of users, who also have strong predictive capability, are neglected as neighbors by algorithms based on vector similarity.

Throughout this work it is evident that exploitation of other types of dependence between ratings can improve recommendations. We explore this idea from two different perspectives: first, as a way to compute predictions, and second, as an alternative to measure similarities between users:

- Related to how the system computes predictions, each neighbor individually proposes a probability distribution representing uncertainties about how the active user could rate the target item. Thus, assuming that a neighbor rates the target item as $r_{u,t}$, the proposed distribution will be based on the active user's ratings for items that have been rated by *this* neighbor as $r_{u,t}$. Finally, all these individual distributions are combined into a final rating recommendation.
- In computing user similarities, we also follow a novel approach based on predictive probabilities. Thus, instead of considering similarities between raw ratings, we use the distance between past ratings of a user and predictions given by candidate neighbors. This idea is motivated by results obtained in a previous study [27] focusing on how probable it is that active user ratings can be generated by considering the suggestions of candidate neighbors.

By combining the two ideas (neighborhood selection and prediction), we obtain a new memory-based collaborative filtering model, which is the main contribution of this paper. Nevertheless, both ideas can be used in isolation by a RS since both neighborhood selection and prediction processes can be considered orthogonal processes in a memory-based RS. Compared to standard neighborhood-based recommenders [16,23,26], our approach shows competitive results for different data sets.

The remainder of the paper is organized as follows. Section 2 outlines the notation used in the paper. Section 3 presents related work on memory-based RSs. Section 4 describes the proposed model by first introducing how individual suggestions can be combined (as a mixture of multinomial distributions) and then showing how the neighborhood is determined. Section 5 presents the experiments and the results obtained. Finally, Section 6 contains concluding remarks.

2. Notation

We first outline the notation used in the paper. In a recommending framework there are a large number m of items or products $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$ and a large set of n users, $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ and for each user, a set of ratings on the quality of certain observed items in $\mathcal{I}$. Among the set of users we highlight the active user, U_a or A, who is the user interacting with the system; a general user is denoted by U or V. Similarly, we highlight item I_t, the target item for which we are making predictions; I denotes a general item. Finally, $\mathcal{R}$ denotes the set of recorded ratings, where $r_{u,i}$ is the rating given by user U to item I. Ratings predicted by the system are denoted as $\hat{r}_{a,t}$. Let $\mathcal{S}$ be the set of possible ratings (e.g., $\mathcal{S} = \{1, \ldots, 5\}$ or $\mathcal{S} = \{like, dislike\}$). When the user does not rate an item, we consider that $r_{u,i} = \emptyset$. To identify the subset of items rated by user U, we use $\mathcal{R}_U$.

Table 1
User × item rating matrix.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_t
Sara	2	2	2	3	3	3	4	4	4	5	5	5	3
Alf	2	2	2	4	4	4	1	1	1	3	3	3	4
Ann	1	1	1	2	2	2	3	3	3	4	4	4	?

3. Related work

RSs have been a very active research topic for two decades [21,35]. During these years, several systems with different objectives and behaviors have been grouped together under the name RS, and therefore it is difficult to provide a single classification for such systems. In this section we review papers related to our problem. Readers interested in a more detailed study of RSs are referred to [20] and references therein.

One of the tasks an RS is designed for is to find a *rating prediction*, that is, given an unseen item, to predict how much a user would like the item. This is the case for the DVD rental provider Netflix: when a user is browsing the list of new releases, the system predicts ratings for every movie to help the user to decide which movie to rent. In this case, the output of the RS is a rating, on a scale from one to five stars, for instance. The second task is to *find good items*: to try to find the best items (and possibly their ratings) to recommend to a given user. In this case, the results are ranked by taking into account predicted ratings. Although both tasks are closely related, it has been proved that there are significant differences between them [17]. In this sense, an algorithm that is uniformly better at predicting ratings does not necessarily have to be better at finding good items (and vice versa).

There are several approaches for the RS information source [1,12], including collaborative, content-based, hybrid, demographic, knowledge-based, and community-based demographic approaches. Our focus is on collaborative filtering, so we consider this type of RS in more detail. In *collaborative filtering* [35] it is assumed that users will be interested in items that users similar to them have rated highly, so groups of people with similar tastes (in terms of item preferences) to those of the active user are identified and items that they have liked are recommended. The similarity in taste of two users is calculated based on similarity in the ratings history of the users or items. According to [11], collaborative RSs can be grouped into memory-based and model-based approaches. Memory-based algorithms use the entire rating matrix $\mathcal{R}$ to make recommendations [36,18,22,23,39,16,15,16,2]. In model-based algorithms, predictions are made by building (offline) an explicit model [9,25,29,8] which is then used (online) to finally recommend a product to users.

Our study uses a memory-based approach in which predictions can be made in two different ways, depending on views of the rating matrix. (1) User-based methods [22,39,16,15] apply an aggregation measure that considers the ratings given by similar users for the same item. The intuition is that if two users U and V have a similar pattern of ratings, then the rating given by U to the target item is likely to be similar to that of V. (2) Item-based approaches [36,18] take into account the similarity between items (two items are similar if they have been rated similarly) and are a good alternative to user-based methods. In this case, predictions are generated by considering the ratings given by the active user to similar items. In both scenarios the set of ratings $\mathcal{R}$ is directly used to predict ratings; in Section 3.1.4 we highlight the advantages of this approach.

3.1. Neighborhood-based RSs

We now describe the three main steps in a neighborhood-based approach [23]: similarity weighting, neighborhood selection and the processes involved in rating prediction.

3.1.1. Similarity measures

The three most common metrics in user-based RSs [19] are:

- Pearson correlation coefficient (PC): This metric measures the degree of association between ratings patterns using a value between −1 and +1. A positive value is evidence of a general trend whereby high ratings by user U are associated with high ratings by V and low ratings by U tend to be associated with low ratings by V (a negative value for the correlation implies the inverse association). PC can be computed as:

$$\mathrm{PC}(U,V) = \frac{\sum_j (r_{u,j} - \bar{r}_u)\cdot(r_{v,j} - \bar{r}_v)}{\sqrt{\sum_j (r_{u,j} - \bar{r}_u)^2 \cdot \sum_j (r_{v,j} - \bar{r}_v)^2}}, \tag{1}$$

where sums over j are applied to those items that both users U and V have rated, i.e. $j \in \mathcal{R}_U \cap \mathcal{R}_V$. If there are no common items between U and V, then $\mathrm{PC}(U, V) = 0$ by default. Furthermore, $\bar{r}_u$ is the average rating value for user U, i.e. $\bar{r}_u = \frac{1}{|\mathcal{R}_U|}\cdot\sum_{I_k\in\mathcal{R}_U} r_{u,k}$.

A related metric is the Spearman rank correlation, which measures similarity between users by considering a ranking over rated items instead of item ratings.

- Cosine measure: This metric defines the similarity between two users as the cosine of the angle between the rating vectors, with values between 0 and 1. A larger value means a higher similarity for the ratings (the two vectors are closer). The cosine similarity of users U and V is defined as:

$$COS(U,V) = \frac{\sum_{i\in\mathcal{R}_U\cap\mathcal{R}_V} r_{u,i} r_{v,i}}{\sqrt{\sum_{i\in\mathcal{R}_U} r_{u,i}^2 \sum_{i\in\mathcal{R}_V} r_{v,i}^2}}. \tag{2}$$

Cosine similarity ignores differences in rating scales between users. This can be taken into account using adjusted cosine similarity, which performs even better than PC for item-based collaborative filtering [31].

- Mean square difference: This is a distance measure (but the inverse can be considered as a similarity metric) that evaluates the distance between two users as the average squared difference between the ratings given by them to the same items, computed as:

$$MSD(U,V) = \frac{\sum_{i \in \mathcal{R}_U \cap \mathcal{R}_V} (r_{u,i} - r_{v,i})^2}{|\mathcal{R}_U \cap \mathcal{R}_V|}. \tag{3}$$

In different empirical analyses to determine the best vector-based similarity measure for computing similarity weights [7,11,22,23,26,16], PC yielded the best accuracy.

3.1.2. Neighbor selection

To make recommendations for target item I_t, only users who have rated this item can be considered. We have to select the best among these users. It has been demonstrated that it is better to set a fixed number of neighbors (potentially in a range from 20 to 60) than to use a similarity weight threshold [23]. Moreover, a significant increase in prediction accuracy is obtained by incorporating a significance weighting by devaluing correlations based on small numbers of co-rated items. In particular, the best-n neighbors with larger values of the following similarity measure, denoted as $N_t(a)$ (the subindex t represents that these neighbors depend on the target item I_t), are used:

$$sim(U,V) = \mathrm{PC}(U,V) \cdot CF, \tag{4}$$

with

$$CF = \begin{cases} 1 & \text{if } x > 50 \\ \frac{x}{50} & \text{otherwise,} \end{cases}$$

where x is the number of items in common.

3.1.3. Computation of predictions

In this section we present different aggregation approaches used in the literature (see [19] for a good review) to compute the predicted rating $\hat{r}_{a,t}$. As stated in Section 1, these methods use the ratings given to I_t by the best-n neighbors. In particular, we consider two approaches that mainly differ in whether the given ratings are normalized or not.

1. Raw ratings. In this case, we consider that users might have different similarity values and a weight that reflects this similarity is added.

$$\hat{r}_{a,t} = \frac{\sum_{V \in N_t(a)} sim(A,V) r_{v,t}}{\sum_{V \in N_t(a)} sim(A,V)}. \tag{5}$$

2. Normalized ratings. In this case, we consider that users may apply different rating scales to quantify the same level of preference for an item. Different normalization strategies can be followed, but the most popular schemes are mean-centering and a Z-score. For the first, normalization involves comparing a user rating to the mean rating. In this case, the predicted rate is:

$$\hat{r}_{a,t} = \bar{r}_a + \frac{\sum_{V \in N_t(a)} (r_{v,t} - \bar{r}_v) \cdot sim(A,V)}{\sum_{V \in N_t(a)} sim(A,V)}. \tag{6}$$

The second also considers the variance in individual rating scales and the predicted rate is computed by dividing the mean-centered rating by the standard deviation, σ:

$$\hat{r}_{a,t} = \bar{r}_a + \sigma_a \frac{\sum_{V \in N_t(a)} (r_{v,t} - \bar{r}_v)/\sigma_v \cdot sim(A,V)}{\sum_{V \in N_t(a)} sim(A,V)}. \tag{7}$$

Use of rating normalization improves the predictions [23,26], although choice of either mean-centering or a Z-score highly depends on the features of the rating matrix $\mathcal{R}$.

3.1.4. Why memory-based approaches are worth applying

In this section we describe the advantages of using a neighborhood-based method. These methods are popular because they are simple and intuitive on a conceptual level and are sufficient for many real-world problems [22,36,31,41]. The other alternative is the use of a model-based approach [25,10,29,30] in which a model is built offline to reflect the relationships

between users and items. To build such a model, several data-mining techniques might be used and the model is learned from the rating matrix $\mathcal{R}$. The model is used (online) to finally recommend the product to users.

In general, commercial RSs use memory-based methods, whereas model-based methods are usually associated with research RSs [15]. For predicting ratings, model-based approaches seem to outperform memory-based approaches [29], but there is strong support indicating that the best accuracy can be obtained when combining different models that complement the shortcomings of each other, as shown by the Netflix Prize competition [9]. For instance, good results have been obtained using model-based methods to learn interpolation weights in neighborhood-based approaches.

There are other factors besides prediction accuracy that are important from the user's perspective, such as neighborhood-based approaches that take advantage of *serendipity* (novelty) or *justifiability* [19]. Related to serendipity, model-based predictions always tend to recommend items closer to the user's (latent) tastes, so it is difficult to recommend an item very different from his usual preferences. This problem is alleviated in neighborhood-based approaches because they capture local associations in the data. In addition, recommendations can be better explained (justified), which helps users to understand the recommendations. In commercial applications, this understanding is finally related to customer loyalty. These models also have some advantages from the perspective of implementation in real systems: they are simple (easy to implement, with fewer parameters to be tuned), efficient (no training is required and recommendations can be computed instantaneously using pre-computed nearest neighbors) and their scalability is less problematic (little affected by constant addition of users, items or ratings). There are other recommendation tasks, such as group recommending [28,33] and e-learning [13,14], for which memory-based approaches represent a good alternative.

4. Prediction-based model

Computation of predictions in a user-based RS is a process whereby each neighbor $V \in N_t(a)$ suggests a rating for target item I_t, denoted by $f_A(V, t)$, and then different suggestions are combined by weighting the contribution of each neighbor by the prediction confidence:

$$\hat{r}_{a,t} \propto \sum_{V \in N_t(a)} conf(A, V) f_A(V, t). \tag{8}$$

Thus, the models in Section 3.1.3 can be viewed as predictive processes by associating the confidence with (normalized) similarity between users, $conf \equiv sim$, in the sense that the greater the similarity between ratings, the greater is the confidence between predictions. In an RS, confidence does not have to be directly related to similarity; for example, in some studies confidence was associated with user trust [4] or knowledge [13]. With respect to the rating suggestions $f_A(V, t)$, the different alternatives in Section 3.1.3 can be seen as using neighbor raw ratings, that is, $f_A(V, t) = r_{v,t}$ in the case of Eq. (5), whereas in Eqs. (6) and (7) these suggestions are normalized by taking into account how both users rated the observed items, using mean-centering and the Z-score $f_A(V,t) = \bar{r}_a + (r_{v,t} - \bar{r}_v)$ and $f_A(V,t) = \bar{r}_a + \sigma_a(r_{v,t} - \bar{r}_v)/\sigma_v$, respectively.

After presenting a predictive point of view for recommendation processes and before describing our approach, we use an illustrative example.

Example. (Motivation for a predictive approach) Consider the set of ratings in Table 2 and suppose the objective is to compute Ann's prediction for target item I_t. Note that this is only a straightforward example involving several simplifications that do not invalidate our argument.

Neighborhood-based methods can be separated into three steps [23].

1. Weight all users/items with respect to the selected similarity criterion. In our example, we consider the PC coefficient. The last column of Table 2 presents PC values for correlation between each user's ratings and those of Ann.
2. Select a subset of users/items to use as a set of predictors. In our example we assume that recommendations are obtained by considering only the top three (TOP-3) neighbors, who are the three users with the highest PC values (Sara, John and Sue).
3. Compute the prediction from a weighted combination of selected neighbor ratings. For illustrative purposes, we assume that the predicted rating is obtained by considering the average rating given to the target item by these neighbors, $f_A(V, t) = r_{v,t}$ (we consider that all the neighbors contribute equally to the recommendations). Therefore, to compute the

Table 2
Measurement of user similarities. The objective is to provide Ann's prediction for target item I_t.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_t	PC
Sara	2	2	2	3	3	3	4	4	4	5	5	5	4	1.00
John	1	2	1	2	3	3	3	3	3	4	4	5	3	0.91
Sue	1	3	1	3	3	3	2	2	4	2	4	4	2	0.51
Bill	1	5	1	5	1	5	1	5	1	5	1	5	1	0.15
Alf	2	2	2	4	4	4	1	1	1	3	3	3	1	0.00
Joe	1	2	3	1	2	3	1	2	3	1	2	3	3	0.00
Nick	4	4	4	3	3	3	2	2	2	1	1	1	2	−1.00
Ann	1	1	1	2	2	2	3	3	3	4	4	4	?	

prediction of Ann's rating, the ratings given by Sara, John and Sue for I_t are used (note that the same users are selected using the cosine measure). Then, in our simplified model, the predicted rating is obtained as $\hat{r}_{a,t} = (4+3+2)/3 = 3$.

In this example, ratings by Alf and Nick were not used according to the correlation-based criterion. Both users show types of dependence that are not properly captured by PC nor by a vector-based similarity measure: Nick seems to be a user with opposite preferences to those of Ann, with a PC value of -1. Whenever Nick increases his ratings, Ann decreases hers. Similarly, although Alf's rating pattern has no (linear) correlation with Ann's, there is strong dependence between them: there is a fixed difference between the similar ratings they assign to different items. For example, if Alf rates an item as 1, Ann always rates it as 3.

If Nick and Alf were selected as Ann's neighbors, use of the particular rating they gave to the target item would probably worsen the prediction. Thus, in our case the predicted rating is computed as $\hat{r}_{a,t} = (4+3+2+\mathbf{2}+\mathbf{1})/5 = 2.4$.

What is the problem? Each neighbor suggests his or her own rating for the target item and not the one expected when considering how both the active user and this neighbor rated common items in the past.[1] Thus, instead of consider Nick and Alf's ratings to I_t (2 and 1, respectively), it could be more favorable to consider a different proposal. In particular, if Alf rates an item as 1, we could expect Ann to rate it as 3. In addition, the same rating would be expected when using Nick's rating, and therefore Ann's prediction could be computed as $\hat{r}_{a,t} = (4+3+2+\mathbf{3}+\mathbf{3})/5 = 3$.

Thus, this example shows that there are situations for which ratings exhibit other types of dependence that are not properly managed by correlation-based recommendation algorithms. Our predictive approach explores these types of dependence between user ratings from two different perspectives: as a new alternative for computing rating predictions for an unobserved item (Section 4.1) and as the criterion for selecting the best neighbors (Section 4.2). Our hypothesis is that if a user was good in predicting past ratings, then his predictions should also be helpful in recommending ratings in the future.

4.1. Selecting the predicted rating

Prediction is a process whereby each neighbor $V \in N_t(a)$ suggests a rating for target item $f_A(V, t)$ and the ratings are then aggregated in a weighted form (Eq. (8)). In this section we present a different approach: instead of considering the raw rating or a deviation from the mean rating, we consider the probability that the active user rates with a value s, $r_{a,t} = s$, given that we know the value of the rating given by V to the target item, say $r_{v,t} = k$, that is, $\Pr(r_{a,t} = s | r_{v,t} = k)$. Our hypothesis is that these distributions can be used to characterize a user's ratings pattern and that they can be estimated from the rating matrix $\mathcal{R}$ by relative counts over the set of items rated by A and V: $\mathcal{R}_A \cap \mathcal{R}_V$.

Example. (Suggesting probabilities) Following the example in Table 2, we might say that $\Pr(\text{Ann} = 3 | \text{Sara} = 4) = 3/3 = 1.0$ and that $\Pr(\text{Ann} = 3 | \text{Bill} = 5) = 1/6 = 0.16$ or $P(\text{Ann} = 4 | \text{Bill} = 5) = 2/6 = 0.33$.

What might be the rating suggested by, for instance, John for the target item $f_{\text{Ann}}(\text{John}, t)$? In this case, we know that John rated the target item as 4, so it seems natural to consider that Ann's ratings will follow the probability distribution $P(\text{Ann} = s | \text{John} = 3)$, $\forall s \in \{1, 2, 3, 4, 5\}$. We denote this probability distribution by means of a vector of values $[p_1, p_2, \ldots, p_5]$, where each p_i represents the probability that the user rates an item with a value of i. Thus, we say that $P(\text{Ann} = s | \text{John} = 3) = [0/5, 2/5, 3/5, 0/5, 0/5] = [0, 0.4, 0.6, 0, 0]$.

It is interesting to note that the neighbor's suggestions are not a rating, but are instead a probability distribution over the rating values. Besides, they do not depend on whether the ratings are positively or negatively correlated nor on the rating scale of the users. For example, suppose that each of Sara's ratings were decreased by one. In this case, Sara would rate the target item as 3 and the probability distribution $\Pr(\text{Ann} = s | \text{Sara} = 3)$ does not change. Similarly, there is no difference in the predictive probabilities if Bill alternates between 2 and 4 instead of 1 and 5 in Table 2. This represents an advantage over the methods in Section 3.1.3, since no normalization is required.

The main drawback of this approach is that the estimations are not reliable when ratings do not co-occur or they are obtained from a few common observations, that is, the size of $\mathcal{R}_A \cap \mathcal{R}_V$ is small. To avoid this situation we consider that, a priori, all the ratings are equally probable. In particular, we use Laplace estimation, which essentially considers that every possible rating has occurred at least once. Therefore, the predictive probabilities are estimated as:

$$\Pr(r_{a,t} = s | r_{v,t} = k) = \frac{n(r_{a,\cdot} = s, r_{v,\cdot} = k) + 1}{n(r_{v,\cdot} = k) + |\mathcal{S}|}, \tag{9}$$

where $n(r_{a,\cdot} = s, r_{v,\cdot} = k)$ represents the number of items in $\mathcal{R}_A \cap \mathcal{R}_V$, with user A rating as s and user V rating as k, and $n(r_{v,\cdot} = k)$ represents the number of times that user V rates with value k. Following the example in Table 2, we can see that Sara rates the target item as 4 and that $\Pr(\text{Ann} = s | \text{Sara} = 4) = [0.125, 0.125, 0.5, 0.125, 0.125]$, with $s \in \{1, \ldots, 5\}$, respectively.

[1] Note that if we were using, for instance, $f_A(V, t) = \bar{r}_a + (r_{v,t} - \bar{r}_v)$, we would have ratings proposed by Nick and Alf of **2.04** and **2.12**, respectively, which again are not the expected suggestions.

Once each of the neighbors' predictive distributions has been estimated, they must be combined into a single distribution. This distribution represents the probability that the active user rates with a given value and can be seen as a mixture of different multinomial distributions [34]:

$$\Pr(r_{a,t} = s) \propto \sum_{V \in N_t(a)} conf(A, V) \Pr(r_{a,t} = s | r_{v,t}), \ \forall s. \tag{10}$$

This combination also represents a difference from classical methods in that an aggregation of a set of fixed values is used. Since the objective is prediction of the rating that the active user will give to I_t, $\hat{r}_{a,t}$, a final decision is necessary. There are several methods for computing this prediction from a probability distribution over ratings. In this paper we use three different alternatives: the most probable, the average, and the median rating using the a posteriori probability in the form:

MP:

$$\hat{r}_{a,t} = \{r | \max\{\Pr(r_{a,t} = r)\}. \tag{11}$$

MED:

$$\hat{r}_{a,t} = \{r | \Pr(r_{a,t} < r) \leqslant 0.5, \ \Pr(r_{a,t} > r) > 0.5\}. \tag{12}$$

AVG:

$$\hat{r}_{a,t} = \sum_{s} \Pr(r_{a,t}) \cdot s. \tag{13}$$

Example. (Predicting ratings) Following on from the previous examples, assume that $\Pr(r_{a,t} = s) = [0.10, 0.15, 0.30, 0.35, 0.10]$, with $s \in \{1, \ldots, 5\}$. The ratings obtained using the different methods are MP = 4, as it is the most probable, MED = $3(0.10 + 0.15 + 0.30 \geqslant 0.5)$ and AVG = $3((0.10 \cdot 1) + (0.15 \cdot 2) + (0.30 \cdot 3) + (0.35 \cdot 4) + (0.10 \cdot 5) = 3.2)$.

4.2. Neighborhood selection using a predictive-based criterion

In neighborhood-based collaborative filtering, predictions rely on the ratings given by similar users in the system since it is assumed that users with similar tastes tend to rate similarly. Different measures have been used (Section 3.1.1) to capture *similarities* between the ratings given by two users to the set of observed items. These similarities are computed considering the set of items co-rated by the candidate neighbor U and the active user A, $\mathcal{R}_A \cap \mathcal{R}_U$. The left-hand side of Fig. 1 illustrates this approach. As stated, these measures perform well when the ratings are highly correlated, but they are not able to capture other types of dependence between ratings that should not be ignored in RS.

We propose a new metric to measure the quality of a neighbor. Our hypothesis is that if user U was good at predicting past ratings by the active user, then his prediction for an unobserved item will also be good. Therefore, users with better predictions for past ratings by the active user shall belong to the neighborhood. The right-hand side of Fig. 1 illustrates this idea, which involves two steps.

1. *Predicting*: For each item rated by the active user, determine the rating that candidate neighbor U would suggest, i.e. $f_A(U, k), \ \forall I_k \in \mathcal{R}_A$. We denote by $\widehat{\mathcal{F}}_A(U) = \{\hat{r}_{a,1}(u), \ldots, \hat{r}_{a,m}(u)\}$ the set of all suggestions (predictions) made by candidate neighbor U for the m items rated by the active user. In Section 4.3 we show in more detail how these predictions can be performed.

2. Determine the quality of the predictions: We have, on the one hand, the predicted ratings $\widehat{\mathcal{F}}_A(U)$ and, on the other hand, the original ratings, $\mathcal{R}_A$. To measure the quality of the predictions, we can use any distance measure that considers the closeness between the ratings sets. Thus, for an item I_i, we have the true rating given by the active user, $r_{a,i}$, and the rating suggested by the candidate neighbor U, $\hat{r}_{a,i}(u)$. When the candidate neighbor is implicit, we will use the term $\hat{r}_{a,i}$ instead of $\hat{r}_{a,i}(u)$. The cost incurred in this prediction can be defined as the magnitude of difference between the ratings, represented by means of a loss function $L(r_{a,i}, \hat{r}_{a,i}(u))$. Then the predictive capability of a user can be defined as the expected loss over all past ratings by the active user:

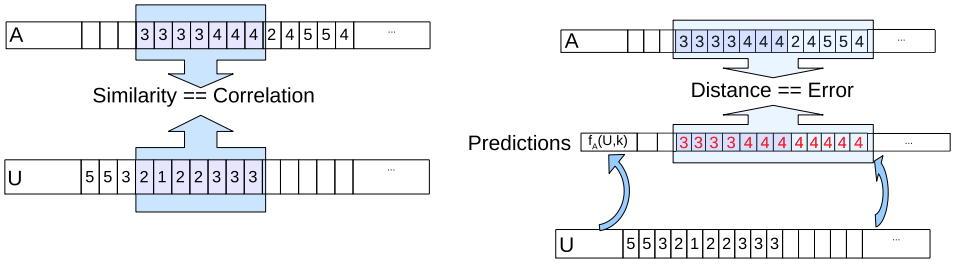

Fig. 1. Correlation- versus predictive-based approach.

$$EL(A,U) = \frac{\sum_{i \in \mathcal{R}_A} L(r_{a,i},\ \hat{r}_{a,i}(u))}{|\mathcal{R}_{\mathcal{A}}|}. \tag{14}$$

We explore two different loss functions related to the Hamming and Manhattan distances and denoted by L_H and L_M, respectively:L_H: This function measures the number of correct predictions, and thus only considers whether a prediction is a success or a failure:

$$L_H(r_{a,i}, \hat{r}_{a,i}) = \begin{cases} 0 & \text{if } r_{a,i} = \hat{r}_{a,i} \\ 1 & \text{otherwise.} \end{cases} \tag{15}$$

L_M: This loss function measures the closeness of predictions to the true ratings by considering the average absolute deviation between a predicted rating, $\hat{r}_{a,i}$, and the user's true rating, $r_{a,i}$:

$$L_M(r_{a,i}, \hat{r}_{a,i}) = abs(r_{a,i} - \hat{r}_{a,i}). \tag{16}$$

For example, considering the data on the right-hand side of Fig. 1, we have $EL_H(A, U) = 3/12 = 0.25$ and $EL_M(A, U) = 3/12 = 0.33$.

Finally, note that correlation-based similarity is not useful for determining prediction quality. For example, strong correlation does not necessarily imply good predictions, as is the case when predictions are equal to the original rating plus a constant value k. Then the ratings are highly correlated (PC = 1) but the predictions worsen with k. Similarly, the cosine measure is not appropriate for this purpose. It does provide an accurate measure of similarity, but with no regard to magnitude, since the rating vectors are treated as unit vectors by normalizing them. However, magnitude is a very important factor in determining prediction quality. For example, suppose $\mathcal{R}_A = \{2, 2, 3, 3\}$ and the suggestions $\widehat{\mathcal{F}}_A(U) = \{1, 2, 3, 4\}$ and $\widehat{\mathcal{F}}_A(V) = \{6, 6, 7, 7\}$, with cosine values 0.967 and 0.993, respectively. In this case, the cosine criterion indicates that V has better predictions than U, which is clearly erroneous.

4.3. Estimating past ratings

The problem we address in this section is how predictions over rated items, $\widehat{\mathcal{F}}_A(U)$, can be computed. A first approach should be use of deviation from the mean rating, $f_A(U, k) = \overline{r_a} + (r_{u,k} - \overline{r_u})$. We have proved that the results obtained with this approach are equivalent to those obtained using a correlation-based neighborhood [3]. Here we explore a different alternative based on the predictive probabilities discussed in Section 4.1, $\Pr(r_{a,i}|r_{u,i})$. Using these probabilities, we can capture different types of dependence among ratings.

Suppose that U is asked about the rating that A could give an item $I_i \in \mathcal{R}_A$, i.e. we want to know $f_A(U, i)$. Obviously, it is assumed that when U has to suggest this rating he does not know the real rating given by A for I_i. Therefore, the information available for making his prediction is $\mathcal{R}_A \setminus \{r_{a,i}\} \cup \mathcal{R}_U$. Two different situations can be considered.

(i) User U rated I_i, i.e. $r_{u,i} \in \mathcal{R}_U$. This is the same situation as when making recommendations for an unobserved item. Thus, the predictive probabilities can be computed using Eq. (9) and then the recommended rating will be obtained as the most probable (Eq. (11)), the median (Eq. (12)) or the average Eq. (13) rating from $\Pr(r_{a,i}|r_{u,i})$.

(ii) User U does not rate I_i, i.e. $r_{u,i} \notin \mathcal{R}_U$. In this case, since U does not rate I_i, a naive approach might be to consider that all the ratings in $\mathcal{S}$ are equally probable. A second alternative might be to use the active user's average rating, $\bar{r}_a$, computed over $\mathcal{R}_A \setminus r_{a,i}$. In both cases, the prediction does not depend on U.

We consider a third alternative in which knowing that U does not rate I_i might be helpful for predicting the rating by A, i.e. we assume a situation in which a missing rating might be informative. For example, consider that A and U only rated 10 of 20 items in common (with scores of 5 and 4, respectively) and that the rest of the 20 ratings given by A were a score of 1. In this case, if we ask U for a suggestion for item I_i, knowing that $I_i$was rated by A and not by U, $f_A(U, i)$, a value of 1 will be selected as the most probable, which differs from the average rating given by A, 3.

Although this is a simplistic example, it demonstrates that there are situations in which the mechanism for obtaining missing ratings cannot be ignored in general. Recent work by Marlin et al. [32] provided evidence that the data typically used for training and testing RSs are missing not at random. This may be a consequence of the fact that users are free to choose which items to rate; for example, a user might have a strong preference for horror movies that is not shared by most users.

Therefore, we again use Laplace estimation to compute the necessary predictive probabilities $\Pr(r_{a,i} = s|r_{u,i} = \emptyset)$, but in this case prior information is taken into account:

$$\Pr(r_{a,i} = s|r_{u,i} = \emptyset) = \frac{n^*(r_{a,\bullet} = s) + \alpha_s}{|\mathcal{R}_A \setminus \mathcal{R}_U| + |\mathcal{S}|}, \tag{17}$$

where $n^*(r_{a,\bullet} = s)$ is the number of items in $\mathcal{R}_A \setminus \mathcal{R}_U$ for which user A gives a rating of s and α_s is the prior probabilities over the candidate ratings computed as the probability that A will give a rating of s from the set $\mathcal{R}_A$ and $|\mathcal{S}|$ denotes the strength of the

prior. Note that if ratings are missing at random, $\Pr(r_{a,i} = s \mid r_{u,i} = \emptyset)$ must be quite similar to $\Pr(r_{a,i} = s)$ and therefore the situation is equivalent to considering only ratings by the active user to make predictions.
Finally, the suggested rating $\hat{r}_{a,i}$ is obtained as the most probable, the median or the average rating.

4.4. Recommendation algorithm and its differences from correlation-based approaches

Once we have presented the two main steps of our methodology, we combine both ideas in a user-based recommendation algorithm able to predict a rating for target item I_t. A sketch of the algorithm is presented in Table 3. The algorithm follows a memory-based approach, and therefore we have to compute $EL(A, U)$ for users U who rated the target item I_t, i.e. U such that $r_{u,t} \in \mathcal{R}_U$. Then these users are ranked in increasing order of their expected loss and the top N are selected as neighbors of the active user.[2] Each neighbor contributes with a probabilistic prediction; these are then combined into a single probability distribution. In the algorithm in Table 3, $wConf(A,V)$ represents the (normalized) weighted contribution of each neighbor.

Now we highlight the main differences with respect to classical user-based similarity approaches. The first is that our approach considers different types of relationship or dependence between ratings beyond linear correlations or scale invariance. To determine the strength of such relationships, we use rating predictions instead of raw ratings. Moreover, to make these predictions, our approach considers that missing ratings might be informative. There are also some differences in the predictive step. For example, on the one hand we combine predictive probabilities instead of fixed suggestions and, on the other hand, by using these probabilities we avoid the problem of rating normalization.

From an efficiency perspective, our approach is more expensive than vector-based measures since predictions for past ratings must be determined to compute the similarity. Nevertheless, it can be computed in time linearly related to the number of items rated by the active user (it is necessary to iterate twice over the rated items to compute the conditional probabilities (step 1.1.1 in Table 3) and to compute the predictions and EL (steps 1.1.2 and 1.1.3 in Table 3)).

5. Experimental results

This section establishes the evaluation settings and presents experimental results for the model. Note that our aim is to evaluate the approach as an alternative to memory-based RSs, which complement and improves the existing approaches. With respect to the experimental methodology, firstly we will show the selected data sets, the evaluation metrics selected, and finally we will discuss the experimental results.

Data sets We used three different data sets, two from film recommendation sites and one from a music recommendation site, which have been divided into training and test sets in such a way that (i) no rating belongs to both, training and test, sets and (ii) there are no test users/items that do not also appear in the training data:

- *1M MovieLens (ML):* The data in this set were collected by the GroupLens Research Project at the University of Minnesota [43] and comprise 1,000,209 anonymous ratings (on a scale from 1 to 5) for approximately 3900 movies by 6040 users who joined MovieLens in 2000. We randomly created a test set using approximately 20% of the ratings for each movie and used the rest of the ratings for the training set. Particularly, the test set contains 192,710 ratings and the training set contains 807,499 ratings. This data set is very commonly used [15,7,6,38], which allows us to benchmark our algorithm against standards in the literature.
- *Yahoo Movies (YMv):* This data set contains ratings for 11,915 movies rated by 7642 users. Training and test sets were provided by Yahoo! [42], and have been obtained by randomly selecting 211,231 and 10,136 ratings, respectively. In this case we used the Yahoo Movies converted ratings (on a scale from 1 to 5) for comparison purposes.
- *Yahoo Music (YMs):* This data set contains ratings for songs supplied by users during normal interaction with Yahoo Music services collected between 2002 and 2006 [42]. The data include approximately 300,000 user-supplied ratings (on a scale from 1 to 5) given by 15,400 users for 1000 songs, with at least ten ratings for each user. We randomly created a test set using approximately 20% of the ratings for each song and used the rest of ratings for the training set. Test and training sets contain 59,250 and 252,454 ratings, respectively.

Evaluation metrics. When evaluating an RS, the metric selected depends on the particular recommendation tasks to be analyzed. According to Ghani and Gunawardana, "the decision on the proper evaluation metric is often critical, as each metric may favor a different algorithm" [17]. Since our aim is to measure the capability of the system to predict user interest in an unobserved item, we focus on the task of *rating prediction*. To test the performance of a model in this task, a widely used approach is to measure the predictive accuracy, i.e. how well the system predicts true ratings or preferences. In this paper, we use three different metrics [24,17].

[2] We also explored the idea of considering the quality of a user as a likelihood function, i.e. we measure how likely the active user's ratings are if we know how the candidate neighbor rates, i.e. $\Pr(\mathcal{R}_A|\mathcal{R}_U)$ [27].

Table 3
Predictive model.

Inputs: active user A, target item I_t
Output: predicted rating $\hat{r}_{a,t}$

1. Neighbor selection
   1.1 For each U who rated the target item I_t
      1.1.1 Compute the conditional probabilities
      $$\Pr(r_a = x | r_u = y), \forall x \in \mathcal{S}, \forall y \in \mathcal{S} \cup \{\emptyset\}$$
      1.1.2 Make the prediction vector $\widehat{\mathcal{F}}_A(U) = \{\hat{r}_{a,1}(u), \ldots, \hat{r}_{a,m}(u)\}$
      1.1.3 Compute $EL(A,U)$.
   1.2 Rank the users and select the best-N neighbors, $N_t(a)$.
2. Predictive process
   2.1 Aggregate the predictive probabilities for each neighbor V
   $$\Pr(r_a = x) = \sum_{V \in N_t(a)} wConf(A,V) \Pr(r_a = x | r_{v,t}), \ \forall x \in \mathcal{S}$$
   2.2 Select a rating $\hat{r}_{a,t}$ from $\Pr(r_a)$.

- Accuracy-based metric
  - MAE measures the distance between predicted and true preferences over items and is the metric typically used for this purpose.[3] MAE is defined as the mean absolute error between the true test ratings, $\mathcal{R}$, and the predicted ratings, $\widehat{\mathcal{R}}$.
- Classification-based metrics
  - %S, percentage of success: this metric focuses on the number of correct predictions given by the system and does not take into account the detailed information provided by the numerical ratings, i.e. the differences between proposed and real ratings.
  - Recall and precision: finally, recall and precision measure the degree to which the system presents relevant information by computing the portion of favored items suggested from the total number of relevant or recommended items. Both metrics are inversely related, such that when precision is increased, recall usually decreases, and vice versa. The F1 metric [F1 = 2 (precision*recall)/(precision + recall)] is used to combine them into a single metric. To measure recall and precision in a regular way [15,40,37], we established a relevance threshold at a value of 4, assuming that an item is relevant if it has been rated positively. Thus, all items rated as 4 and 5 represent a relevant statement, whereas all ratings of 1, 2 and 3 imply a non-relevant statement.

Baseline. In this paper we use as baseline the standard user-based algorithm, developed by Herlocker et al. [23], for which neighbors are selected using PC with a significant weight correction according to Eq. (4) and the individual suggestions are combined using mean-centering according to Eq. (6). This works well with different data sets and is the most commonly used approach [22,23,7,15,40]; therefore, any alternative methods proposed must showed improved results in comparison to this baseline.

5.1. Experimental results

Before the results are discussed, some considerations are in order. First, our model computes a probability distribution over candidate ratings, and this probability has to be converted to a rating, $\hat{r}$, in the range from 1 to 5. The criterion used to obtain $\hat{r}$ is related to the accuracy metric we want to optimize: if we predict the most probable rating, we minimize %S, and if we predict the median rating, we minimize MAE. Second, and also related to the accuracy criterion, the loss function used to learn the neighborhood depends of the criterion we want to evaluate: when considering MAE or %S, we learn using the loss function EL_M or EL_H, respectively.[4]

Correlation versus predictive approaches. To compare correlation- and predictive-based approaches, we must first check whether the learned neighborhoods are similar. This is important because of the effect of the neighborhood on predictions. If the neighborhoods are similar, differences between the approaches can be attributed to the method for combining the suggestions of neighbors. Table 4 shows the percentage of common neighbors between the two approaches when considering the mean average error as the loss function. We can conclude, on the one hand, that quite different neighborhoods were learned and, on the other hand, that these differences are database-dependent. It is evident that the Yahoo and MovieLens databases behave differently. The two neighborhood selection criteria show greater agreement for Yahoo Movies (particularly for a small neighborhood) than for MovieLens.

[3] Another accuracy metric is the root mean squared error (RMSE), which gives greater weight to higher errors, and computes the root of the expected squared loss to obtain the error for ratings of the same magnitude. It has been proved that this metric is correlated with MAE.

[4] Our experiments proved that worse results are obtained with other combinations.

Table 4
Number of common neighbors: baseline versus expected loss.

	5	10	20	30	50
MovieLens	29.59	34.77	39.81	41.31	38.41
Yahoo movie	75.37	65.49	55.86	50.15	42.60
Yahoo music	64.47	61.35	54.48	48.33	39.98

Table 5 shows MAE and %S for the selected data sets (the precision and recall results presented in Fig. 2 are discussed later) obtained when using a pure correlation-based approach (our baseline approach) and a pure predictive-based approach with the algorithm in Table 3, where *wConf* in step 2.1 is computed by considering the normalized expected loss over past predictions, denoted as *wM* and *wH* in Table 5. Some conclusions can be drawn from the results. The first is that our predictive proposals are competitive (our best results outperform (5/6) those obtained with the baseline). This confirms our initial hypothesis that many users who might be good for predictive purposes are not captured by correlation criteria.

In preliminary experiments, we found that use of any criterion to give a higher weight to neighbors with a large number of co-rated items did not improve neighborhood quality. We believe that when both users have few co-rated items, the predictions for the set of items rated by the active user, $\mathcal{R}_A$, are not good enough, and therefore these neighbors are naturally rejected by the selection process. For example, if two users rate only one item in common, then the ratings suggested by the candidate neighbor will be always the same, close to the mean rating, and the predictive capability can easily be improved by including other users in the system. We also investigated the effect of considering a weighting for the number of co-rated items (*wConf*) when combining the predictive probabilities with the normalized expected loss, but the results obtained were not conclusive. These two facts are important, since they indicate that our predictive approach represents a robust selection criterion.

Modifying the confidence criterion wConf. Our previous experiment considered that confidence values for the suggestions *wConf* are directly related to the criteria used to learn the neighborhood. Although this can be a good choice when designing a "pure" recommendation system, this decision is not the most appropriate for a base. Since correlation and predictive capabilities measure two different types of dependence between user ratings, we explore whether the combination of these two criteria can improve recommendations.

We also used the weighted expected loss when considering a correlation-based approach, but the results were quite similar to those obtained using weighted Pearson values and thus are not presented here. Our guess is that highly correlated users present good suggestions and the weighted expected loss preserves the relative importance among different neighbors. However, good results were obtained when using correlation-based confidences combined with predictive probabilities. We had to modify the standard correlation-based weights since users with opposite tastes can exist in the neighborhood. Thus, confidence depends on the one hand on how ratings are correlated, in absolute terms and, on the other hand, on the ratio of common ratings:

$$wConf(A,U) = \frac{abs(\mathrm{PC}(A,U)) \times |\mathcal{R}_A \cap \mathcal{R}_U|/|\mathcal{R}_A|}{\sum_{V \in N_t(a)} abs(\mathrm{PC}(A,V)) \times |\mathcal{R}_A \cap \mathcal{R}_V|/|\mathcal{R}_A|}. \tag{18}$$

In our experiments this correction factor gave better results than the one used in Eq. (4). Table 6 shows the results obtained, from which some conclusions can be drawn. First, better results were obtained by combining predictive capability and rating correlations, which indicates that the two approaches reinforce each other. Second, the number of neighbors required to obtain the best prediction (neighborhood size) has been reduced considerably (surprisingly, with only 10 neighbors we obtain good predictions with all the metrics); this demonstrates the ability of the combination to find the very best neighbors.

Recall and Precision metrics. Now we discuss the recall and precision results obtained with the different models. Fig. 2 shows the precision (P), F1 and recall (R) values obtained for each data set and for different neighborhood sizes using a pure correlation-based approach (basel), a pure predictive approach using the Hamming (ELH) or Manhattan (ELM) distance, or a predictive approach considering Eq. (18) as the confidence criterion (ELH-wPr and ELM-wPr). A first conclusion is that a correlation-based approach improves precision, whereas a predictive approach improves recall; this holds for all the data sets. For the F1 metric, which combines both measures, we can say that a predictive criterion outperforms a correlation-based approach. In addition, it is evident that the effect of neighborhood size differs for the data sets. In general, we can say that a small neighborhood size increases (decreases) precision (recall) for MovieLens, but the trend is reversed for Yahoo Music and Yahoo Movies, which exhibit more erratic performance. The best model in terms of the F1 metric depends on the data set. The best results were obtained with ELH for MovieLens (F1 = 0.7979), with ELH-wPr for Yahoo Movies (F1 = 0.8915) and with ELM-wPr for Yahoo Music (F1 = 0.6638); these results represent improvements with respect to the baseline of 21%, 10%, and 14%, respectively.

Thus, the method for computing confidence is an important factor. Moreover, it has been shown that this confidence is not related to the way in which the neighborhood is learned (this has also been demonstrated, for instance, in trust-based RSs, for which user confidence is related to a trust criterion [4], or in e-learning systems where the users with greater

Table 5
Correlation versus predictive approach.

Neighbors	5	10	20	30	50
MAE metric MovieLens data set					
Baseline	0.7464	0.7191	0.7077	0.7057	**0.7047**
$EL_M + wM$	0.7164	0.7081	0.7036	0.7014	**0.6967**
MAE metric yahoo movies					
Baseline	0,7530	**0.7453**	0.7477	0.7502	0.7543
$EL_M + wM$	0.7476	**0.7439**	0.7555	0.7606	0.7687
MAE metric yahoo music					
Baseline	0,9701	0,9377	0,9212	**0,9181**	1,1235
$EL_M + wM$	1,0600	1,0491	1,0385	1,0289	**1,0109**
%S metric MovieLens data set					
Baseline	42.731	43.903	44.380	44.559	**44.570**
$EL_H + wH$	43.409	44.108	44.240	44.550	**44.581**
%S metric yahoo movies					
Baseline	45,9747	46,9021	47,0501	**47,4152**	47,4053
$EL_H + wH$	52,3086	53,3248	**53,7194**	53,6997	53,6504
%S metric yahoo music					
Baseline	35,04	35,29	35,39	**35,42**	35,14
$EL_H + wH$	39,5781	40,7105	41,1409	**41,3215**	40,8017

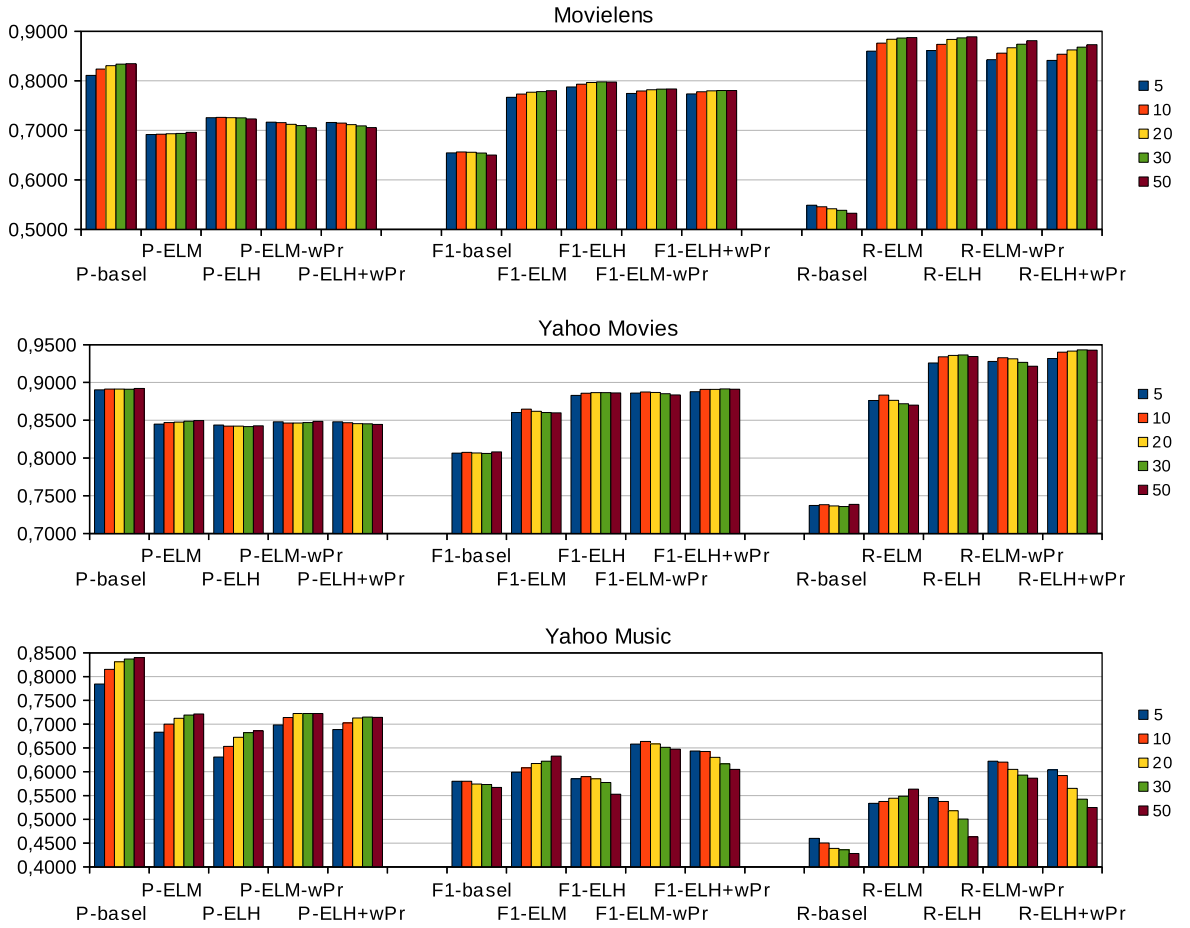

Fig. 2. Recall and precision values for the different data sets.

knowledge have greater weights [13]); improvements in MAE values of approximately 3.5%, 8.5%, and 11.5% for MovieLens, Yahoo Movies and Yahoo Music, respectively, support this claim.

Table 6
Combination of correlation and predictive capabilities.

Neighbors	5	10	20	30	50
MAE metric $EL_M + wPr$					
MovieLens	0.6843	**0.6798**	0.6799	0.6805	0.6842
Yahoo movies	0.6849	**0.6804**	0.6808	0.6879	0.6934
Yahoo music	0,9297	**0,9278**	0,9465	0,9660	0,9795
%S metric $EL_H + wPr$					
MovieLens	44.480	44.794	**44.824**	44.676	44.378
Yahoo movies	53,4530	54,0549	54,0746	**54,2423**	54,0943
Yahoo music	47,5190	**47,6844**	46,8658	45,7165	44,6970

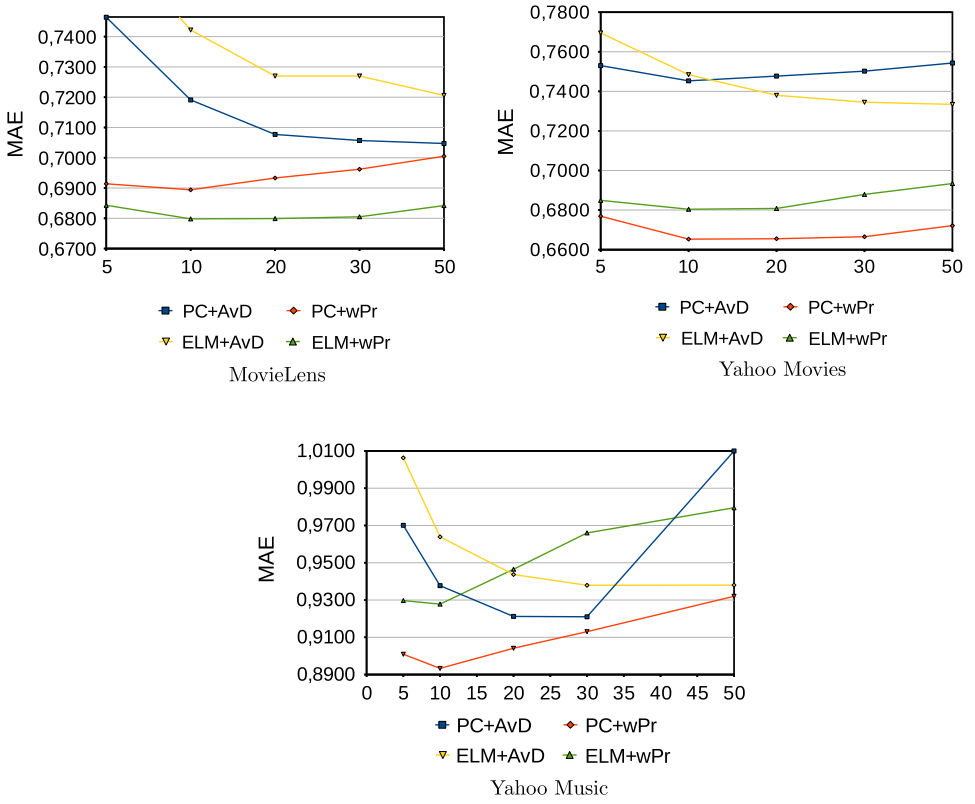

Fig. 3. Combining different recommendation strategies.

Finally, we compare the results with a state-of the art approach. Kernel-based systems [5] have yielded very good results for different data sets and are among the state-of-art approaches for MovieLens. We use the results reported by Ghazanfar et al. even though this is less than ideal. The authors reported MAE values of 0.687 (user-based kernel approach), 0.668 (item-based kernel approach) and 0.664 (hybrid-based kernel approach), which represent improvements compared to the baseline MAE of 0.715 (user-based collaborative filtering) of 3.9%, 6.5%, and 7.1%, respectively. These results illustrate the quality of our proposal, particularly if we focus on the user-based kernel approach, since this uses the same information as our methods, and achieves similar improvements with respect to the baseline.

5.1.1. Combining the approaches

After considering the previous section, the following question arises: could neighborhood selection (*N_Sel*) and rating prediction (*Pred*) be considered orthogonal processes in an RS? Obviously, from an algorithmic perspective the answer is yes. For example, we can learn the neighborhood with a correlation criterion and use predictive processes to compute the rating prediction. The question we consider in this section is whether these types of combination are good or not in terms of the accuracy of the results.

Table 7
Best experimental results.

Data set	MAE	*NN*	*N_Sel*	Pred.
MovieLens	0.6798	10	EL	wPr
Yahoo movies	0.6653	10	PC	wPr
Yahoo music	0.8933	10	PC	wPr

Fig. 3 shows the MAE values obtained when combining different strategies for these tasks.[5] The prefixes PC and ELM denote whether neighbors were selected by maximizing weighted PC (Eq. (4)) or minimizing the expected loss in the predictions (Eq. (16)). For combining the suggestions of neighbors in the prediction step, suffix AvD denotes that the prediction was obtained by combining differences with respect to the average rating (Eq. (6)) and wPr denotes that the user suggestions are weighted predictive probabilities (Step 2.1 of the algorithm in Table 3, with the weights in Eq. (18)). Table 7 summarizes the best results, where *NN* is the optimum neighborhood size.

From these results some conclusions can be drawn. First, the experimentation shows that the best results were obtained by combining predictive probabilities in the prediction process. This implies that combining predictive probabilities is a robust strategy, independent of the criterion used to select the neighborhood. Thus, combining correlation and predictive probability approaches seems to be an alternative worth considering. The results represent improvements of 3.53%, 8.71%, and 2.70% compared to the pure correlation approach (baseline) for MovieLens, Yahoo Movies and Yahoo Music, respectively (statistically significant using the adjusted t-test at a significance level of 0.05). The poor results of the ELM + AvD combination (particularly compared with ELM + wPr) is remarkable. We hypothesize that this situation can be explained by selected neighbors whose ratings do not correlate with those of the active user, and therefore use of AvD is not necessarily a good approach. With respect to neighborhood size, an important conclusion is that when using *wPr* as a combination strategy the optimal number of neighbors is approximately 10, whereas AvD requires large values. Finally, and as global conclusion, this paper shows that there are neighbors with good predictive capability that are neglected by correlation-based approaches and that the information they provide can be used to improve RS performance.

6. Conclusions

We have shown that the use of predictive probabilities (how probable it is that the active user will rate a target item with a value given that we know the ratings given by his neighbors) is a valid strategy for both identifying the neighborhood and combining individual suggestions. Using a predictive-based approach, we found that many users who might be good for predictive purposes are not captured by classical correlation criteria. Moreover, in agreement with the lessons learned from the Netflix competition, experiments revealed that an effective way to achieve a better recommendation accuracy for different data sets is to combine different approaches, such as correlations and predictive capabilities in our case.

In future work, we will explore how the proposed technique can be integrated with model-based approaches, for instance the learned model can be used to learn interpolation weights while using neighborhood, independently of the criteria used to select this neighborhood. Also, we are working on studying different mechanisms to incorporate additional knowledge in order to give a context-based strength to the individual suggestions. By means of this knowledge a user might be considered a good candidate for predicting the rating for some items, for instance sports news or horror movies, whereas the same user can be discarded when suggesting the rating for economy news or family movies. Similarly, by means of this context we can represent that new ratings might be considered more important than the old ones.

Acknowledgments

This work was jointly supported by the Spanish Ministerio de Educación y Ciencia (TIN2008-06566-C04-01 and TIN2011-28538-C02-02), Spanish Research Program Consolider Ingenio 2010: MIPRCV (CSD2007-00018) and Junta de Andalucía Excellence Project TIC-04526.

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749.
[2] H.J. Ahn, A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem, Information Sciences 178 (2008) 37–51.
[3] S. Cleger-Tamayo, J.M. Fernández-Luna, J.F. Huete, A new criteria for selecting neighborhood in memory-based recommender systems advances in artificial intelligence, in: 14th Conference of the Spanish Association for Artificial Intelligence, CAEPIA'11, 2011, pp. 423–432.
[4] J. O'Donovan, B. Smyth, Trust in recommender systems, in: 10th International Conference of Intelligent User Interfaces, 2005, pp. 167–174.
[5] M. Ghazanfar, A. Prugel-Bennett, S. Szedmak, Kernel Mapping Recommender System algorithms Information Sciences <http://eprints.ecs.soton.ac.uk/22686/>, 2011.

[5] We only show MAE values because it is the metric typically used to measure the accuracy of these systems, but the results obtained when considering %S is similar.

[6] M.G. Vozalis, K.G. Margaritis, Using SVD and demographic data for the enhancement of generalized collaborative filtering, Information Sciences 177 (15) (2007) 3017–3037.
[7] L. Candillier, F. Meyer, M. Boulle, Comparing state-of-the-art collaborative filtering systems, in: Proceedings of 5th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM 2007, LNAI 4571, 2007, pp. 548–562.
[8] S. Cleger-Tamayo, J.M. Fernández-Luna, J.F. Huete, Top-N news recommendations in digital newspapers, Knowledge-Based Systems (2011), http://dx.doi.org/10.1016/j.knosys.2011.11.017.
[9] R. Bell, Y. Koren, Lessons from the netflix prize challenge, SIGKDD Explorations 9 (2007) 75–79.
[10] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, Journal of Machine Learning Research 3 (2003) 993–1022.
[11] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, In: 14th Conference on Uncertainty in Artificial Intelligence, 1998, pp. 43–52.
[12] R. Burke, Hybrid recommender systems: survey and experiments, User Modeling and User-Adapted Interaction 12 (4) (2002) 331–370.
[13] J. Bobadilla, F. Serradilla, A. Hernando, Collaborative filtering adapted to recommender systems of e-learning, Knowledge-Based Systems 22 (4) (2009) 261–265.
[14] C. Porcel, E. Herrera-Viedma, Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries, Knowledge-Based Systems 23 (1) (2010) 32–39.
[15] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowledge-Based Systems 23 (6) (2010) 520–528.
[16] F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender systems, ACM Transactions on the Web 5 (1) (2011) 2:1–2:33.
[17] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, Journal of Machine Learning Research 10 (2009) 2935–2962.
[18] M. Deshpande, G. Karypis, Item-based top-N recommendation algorithms, ACM Transactions on Information Systems 22 (1) (2004) 143–177.
[19] C. Desrosiers, G. Karypis, A comprehensive survey of neighborhood-based recommendations methods, in: F. Ricci, L. Rokach, B. Shapira, P. Kantor (Eds.), Recommender System Handbook, Springer, 2011, pp. 107–144.
[20] F. Ricci, L. Rokach, B. Shapira, P. Kantor (Eds.), Recommender System Handbook, Springer, 2011.
[21] D. Goldberg, D. Nichols, B.M. Oki, D. Terry, Using collaborative filtering to weave an information tapestry, Communications of the ACM 35 (12) (1992) 61–70.
[22] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the ACM SIGIR 99, 1999, pp. 230–237.
[23] J.L. Herlocker, J.A. Konstan, J.T. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Information Retrieval 5 (4) (2002) 287–310.
[24] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.
[25] T. Hofmann, Latent semantic models for collaborative filtering, ACM TOIS V22 (1) (2004) 89–115.
[26] A.E. Howe, R.D. Forbes, Re-considering neighborhood-based collaborative filtering parameters in the context of new data, in: Proceedings of the 17th ACM conference on Information and knowledge management, CIKM'08, 2008, pp. 1481–1482.
[27] L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, M.A. Rueda-Morales, Measuring predictive capability in collaborative filtering RecSys '09, in: Third ACM Conference on Recommender Systems, 2009, pp. 313–316.
[28] L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, M.A. Rueda-Morales, Managing uncertainty in group recommending processes, User Modeling and User-Adapted Interaction 19 (3) (2009) 207–242.
[29] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer 42 (8) (2009) 30–37.
[30] Y. Koren, R. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira, P. Kantor (Eds.), Recommender System Handbook, Springer, 2011, pp. 145–186.
[31] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Computing (2003) 76–80.
[32] B. Marlin, R. Zemel, Collaborative prediction and ranking with non-random missing data, in: ACM Conference on Recommender Systems (RecSys), 2009.
[33] J. Masthoff, Group recommender system: combining individual models, in: F. Ricci, L. Rokach (Eds.), Recommender Systems Handbook, Springer, 2011, pp. 677–704.
[34] G.L. McLachlan, D. Peel, Finite Mixture Models, Wiley, 2000.
[35] P. Reskick, H.R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 56–58.
[36] B. Sarwar, G. Karypis, J. Konstan, J. Reidl. Item based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on WWW, 2001, pp. 285–295.
[37] T. Jambor, J. Wang, Goal-driven collaborative filtering a directional error based approach, in: ECIR, Springer, 2010.
[38] P. Cremonesi, Y. Koren, R. Turri, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the Fourth ACM Conference on Recommender Systems, 2010, pp. 339–346.
[39] U. Shardanand, P. Maes, Social information filtering: algorithms for automating "word of mouth", in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 1995, pp. 210–217.
[40] A. Bellogín, P. Castells, I. Cantador, Precision-oriented evaluation of recommender systems: an algorithmic comparison, in: Proceedings of the Fifth ACM Conference on Recommender Systems, 2011, pp. 333–336.
[41] J. Wang, A.P. de Vries, M. Reinders, Unified Relevance Models for Rating Prediction in Collaborative Filtering ACM TOIS 26 (2008) 3. Article 16.
[42] Yahoo! Webscope dataset <http://research.yahoo.com/Academic_Relations>.
[43] Movielens Data Sets <http://www.grouplens.org/>.