A collaborative filtering approach to mitigate the new user cold start problem

Jesús Bobadilla , Fernando Ortega, Antonio Hernando, Jesús Bernal

A B S T R A C T

The new user cold start issue represents a serious problem in recommender systems as it can lead to the loss of new users who decide to stop using the system due to the lack of accuracy in the recommendations received in that first stage in which they have not yet cast a significant number of votes with which to feed the recommender system's collaborative filtering core. For this reason it is particularly important to design new similarity metrics which provide greater precision in the results offered to users who have cast few votes. This paper presents a new similarity measure perfected using optimization based on neural learning, which exceeds the best results obtained with current metrics. The metric has been tested on the Netflix and Movielens databases, obtaining important improvements in the measures of accuracy, precision and recall when applied to new user cold start situations. The paper includes the mathematical formalization describing how to obtain the main quality measures of a recommender system using leave-one-out cross validation.

1. Introduction

Recommender systems (RS) [48] enable recommendations to be made to users of a system in reference to the items or elements on which this system is based (books, electrical appliances, films, e-learning material, etc.). The core of a RS lie in its filtering algorithms: demographic filtering [26] and content-based filtering [28,47] are two well known filtering techniques. Content-based RS base the recommendations made to a user on the choices this user has made in the past (e.g. in a web-based e-commerce RS, if the user purchased computer science books in the past, the RS will probably recommend a recent computer science book that he has not yet purchased on this website); demographic filtering based RS are based on the assumption that individuals sharing certain common personal features (sex, age, country, etc.) will also share common preferences.

Currently, collaborative filtering (CF) is the most commonly used and studied technology [1,17]. CF RS are based on the way in which humans have made decisions throughout history: in addition to our own personal experience, we also base our decisions on the experiences and knowledge coming from a relatively large group of acquaintances. We take this set of knowledge and we consider it "in a critical way", to obtain the decision we think will best suit our goal. "In a critical way" means that we are more inclined to take into consideration those suggestions made by people with whom we have more in common regarding the pursued goal; for instance, before throwing ourselves down a steep slope on a small snowboard, we listen to all of our friends' opinions, but we regard much highly those that we consider to have more in common with our level in that sport, with our liking for risk, etc.

There are also some so-called hybrid RS: they are systems which combine different filtering approaches to exploit merits of each one of these techniques, such as a combination of CF with demographic filtering or CF with content based filtering [2]. Among the natural fields of action of these hybrid models we can highlight some bio-inspired models which are used in the filtering stage [14].

CF based RS allow users to give ratings about a set of elements (e.g. hotels, restaurants, tourist destinations, etc. in a CF based website), in such a way that when enough information is stored on the system we can make recommendations to each user based on information provided by those users we consider to have the most in common with them. Movie recommendation websites are probably the best-known cases to users and are without a doubt the most thoroughly studied by researchers [25,2], although there are many other fields in which RS have great and increasing importance, such as e-commerce [23,57,52,36], e-learning [12,4,21], music [15,31], digital libraries [40], playing games [32], social networks based collaborative filtering [13,35], etc.

A key factor in the quality of the recommendations obtained in a CF based RS lies in its capacity to determine which users have the most in common (are the most similar) to a given user. A series of algorithms [19] and metrics [1,5,55,7–9] of similarity between users are currently available, enabling this important function to be performed in the CF core of this type of RS.

In order to measure the quality of the results of a RS, there is a wide range of metrics which are used to evaluate both the prediction and recommendation quality of these systems [17,1,18,6].

CF based RS estimate the value of an item not voted by a user via the ratings made on that item by a set of similar users. The overall quality in the prediction is called accuracy [3] and the mean absolute error (MAE) is normally used to obtain it [17]. The system's ability to make estimations is called coverage and it indicates the percentage of prediction which we can make using the set of similar users selected (usually, the more similar users we select and the more votes the selected users have cast, the better the coverage we achieve). In RS, besides aiming to improve the quality measures of the predictions (accuracy and coverage), there are other issues that need be taken into account [56,41,51]: avoiding overspecialization phenomena, finding good items, credibility of recommendations, precision and recall measures, etc.

The rest of the paper is divided into the following sections (with the same numbering shown here):

2. State of the art, in which a review is made of the most relevant contributions that exist in the CF aspects covered in the paper: cold-start and application of neural networks to the RS.
3. General hypothesis and motivations: what we aim to contribute and the indications that lead us to believe that carrying out research into this subject will provide satisfactory results that support the hypothesis set out.
4. Design of the user cold-start similarity measure: explanation and formalization of the design of the similarity measure proposed as a linear combination of simple similarity measures, by adjusting the weights using optimization techniques based on neural networks.
5. Collaborative filtering specifications: formalization of the CF methodology which specifies the way to predict and recommend, as well as to obtain the quality values of the predictions and recommendations. This is the formalization that supports the design of experiments carried out in the paper. The methodology is provided which describes the use of leave-one-out cross validation applied to obtaining the MAE, coverage, precision and recall.
6. Design of the experiments with which the quality results are obtained provided by the user cold-star similarity measure proposed and by a set of current similarity measures for which we aim to improve the results. We use the Netflix (http://www.netflixprize.com) and Movielens (http://www.movielens.org) databases.
7. Graphical results obtained in the experiments, complemented with explanations of the behavior of each quality measure.
8. Most relevant conclusions obtained.

2. State of the art

2.1. The cold-start issue

The cold-start problem [48,1] occurs when it is not possible to make reliable recommendations due to an initial lack of ratings. We can distinguish three kinds of cold-start problems: new community, new item and new user. The last kind is the most important in RS that are already in operation and it is the one covered in this paper.

The new community problem [49,27] refers to the difficulty in obtaining, when starting up a RS, a sufficient amount of data (ratings) which enable reliable recommendations to be made. When there are not enough users in particular and votes in general, it is difficult to maintain new users, which come across a RS with contents but no precise recommendations. The most common ways of tackling the problem are encouraging votes to be made via other means or not making CF-based recommendations until there are enough users and votes.

The new item problem [38,39] arises due to the fact that the new items entered in RS do not usually have initial votes, and therefore, they are not likely to be recommended. In turn, an item that is not recommended goes unnoticed by a large part of the users community, and as they are unaware of it they do not rate it; in this way, we can enter a vicious circle in which a set of items of the RS are left out of the votes/recommendations process. The new item problem has less of an impact on RS in which the items can be discovered via other means (e.g. movies) than in RS where this is not the case (e.g. e-commerce, blogs, photos, videos, etc.). A common solution to this problem is to have a set of motivated users who are responsible for rating each new item in the system.

The new user problem [42,43,46] is among the great difficulties faced by the RS in operation. When users register they have not cast any votes yet and, therefore, they cannot receive any personalized recommendations based on CF; when the users enter their firsts ratings they expect the RS to offer them personalized recommendations, but the number of votes entered is usually not sufficient yet to provide reliable CF-based recommendations, and, therefore, new users may feel that the RS does not offer the service they expected and they may stop using it.

The common strategy to tackle the new user problem consists of turning to additional information to the set of votes in order to be able to make recommendations based on the data available for each user; this approach has provided a line of research papers based on hybrid systems (usually CF-content based RS and CF-demographic based RS). Next we analyze some hybrid approaches; [30] propose a new content-based hybrid approach that makes use of cross-level association rules to integrate content information about domains items. Kim et al. [24] use collaborative tagging employed as an approach in order to grasp and filter users' preferences for items and they explore the advantages of the collaborative tagging for data sparseness and a cold-start user (they collected the dataset by crawling the collaborative tagging del.icio.us site). Weng et al. [53] combine the implicit relations between users' items preferences and the additional taxonomic preferences so as to make better quality recommendations as well as alleviate the cold-start problem. Loh et al. [33] represent user's profiles with information extracted from their scientific publications. Martinez et al. [34] present a hybrid RS which combines a CF algorithm with a knowledge-based one. [10] propose a number of common terms/term frequency (NCT/TF) CF algorithm based on demographic vector. Saranya and Atsuhiro [50] propose a hybrid RS that makes use of latent features extracted from items represented by a multi-attributed record using a probabilistic model. Park et al. [37] propose a new approach: they use filterbots, and surrogate users that rate items based only on user or item attributes.

All the former approaches base their strategies on the presence of additional data to the actual votes (user's profiles, user's tags, user's publications, etc.). The main problem is that not all RS databases possess this information, or else it is not considered sufficiently reliable, complete or representative.

There are so far two research papers which deal with the cold-start problem through the users' ratings information: Hyung [22] presents a heuristic similarity measure named PIP, that outperforms the traditional statistical similarity measures (Pearson correlation, cosine, etc.); and Heung et al. [16] proposes a method that first predicts actual ratings and subsequently identifies prediction errors for each user; taking into account this error information, some specific "error-reflected" models are designed.

The strength of the approach presented in this paper (and in the Hyung and Heung works) lies in its ability to mitigate the new user cold-start problem from the actual core of the CF stage, providing a similarity metric between users specially designed for this purpose and which can be applied to new users of any RS; i.e. it has a universal scope as it does not require additional data to the actual votes cast. The main problem of this approach is that with it, it is more complex and risky to carry out an information retrieval from the votes than to directly take the additional information provided by the user's profiles, user's tags, etc., held by some RS.

2.2. Neural networks applied to recommender systems

Neural networks (NN) is a model inspired by biological neurons. This model, intended to simulate the way the brain processes information, enables the computer to "learn" to a certain degree. A neural network typically consists of a number of interconnected nodes. Each node handles a designated sphere of knowledge, and has several inputs from the network. Based on the inputs it gets, a node can "learn" about the relationships between sets of data, pattern, and, based upon operational feedback, are molded into the pattern required to generate the required results.

In this paper, we make novel use of NN, by using them to optimize the results provided by the similarity measure designed. This approach enables NN techniques to be applied in the same kernel of the CF stage. The most relevant research available in which NN are used in some aspect of the operation of RS usually focuses on hybrid RS in which NN are used for learn users profiles; NN have also been used in the clustering processes of some RS.

The hybrid approaches enable neural networks to act on the additional information to the votes. In [44] a hybrid recommender approach is proposed using Widrow–Hoff [54] algorithm to learn each user's profile from the contents of rated items, to improve the granularity of the user profiling. In [11] a combination of content-based and CF is used in order to construct a system providing more precise recommendations concerning movies. In [29] first, all users are segmented by demographic characteristics and users in each segment are clustered according to the preference of items using the Self-Organizing Map (SOM) NN. Kohonon's SOMs are a type of unsupervised learning; their goal is to discover some underlying structure of the data.

Two alternative NN uses are presented in [20,45]. In the first case the strategy is based on training a back-propagation NN with association rules that are mined from a transactional database; in the second case they propose a model that combines a CF algorithm with two machine learning processes: SOM and Case Based Reasoning (CBR) by changing an unsupervised clustering problem into a supervised user preference reasoning problem.

3. Hypothesis and motivation

The paper's hypothesis deals with the possibility of establishing a similarity measure especially adapted to mitigate the new user cold-start problem which occurs in CF-based RS; furthermore, during the recommendation process, the similarity measure designed must only use the local information available: the votes cast by each pair of users to be compared.

The main idea of our paper considers that it is possible to obtain additional information to that used by the traditional similarity measures of statistical origin (Pearson correlation, cosine, Spearman rank correlation, etc.). Whilst the traditional similarity measures only use the numerical information of the votes, we will make use of both the numerical information of the votes and information based on the distribution and on the number of votes cast by each pair of users to be compared.

As regards the number of votes of the users compared by the similarity measure that we want to design, the key aspect is that it is more reasonable to assign greater similarity to users who have voted for a similar number of items than users for whom the number of items voted is very different. By way of example, it is more convincing to determine as similar users two people who have only voted for between 8 and 14 movies, all of which are science fiction, than to determine as similar users a user who has only voted for 10 movies, all of which are science fiction, and another who has voted for 2400 movies of all genres. Whilst in the first case, the recommendations will tend to be restricted to the movies of common genre, in the second, the new cold-start user will be able to receive thousands of recommendations of all types of movies of genres in which they are not interested and which are very unsuitable for recommending based only on a maximum of 10 recommendations in common.

In the previous example, and under the restriction of recommendations made to new cold-start users, we can see the positive aspect of being recommended by a person who has cast a similar number of votes to yours: it is quite probable that the items that they have voted for and that you have not, will be related to those you have rated, and, therefore, it is very possible that you are interested in them. We can also see the negative aspect: the capacity for recommendation (coverage) of the other user will not be as high.

As regards the distribution (or structure) of the votes of the users compared with the similarity measure we wish to design, there are two significant aspects for determining their similarity:

- It is positive a large number of common items that they have both voted for.
- It is negative a large number of uncommon items that they have both voted for.

If a user $u1$ has voted for 10 movies, a user $u2$ has voted for 14 and a user $u3$ has voted for 70, it will be more convincing to classify $u1$ and $u2$ as similar if they have 6 movies in common than if they have only one. It will also be more convincing to classify $u1$ and $u2$ as similar with 6 movies in common than $u1$ and $u3$ with 7 movies in common. In this case, we must also be cautious with the coverage reached by the similarity measure designed.

The assumptions on which the paper's motivation is based will be confirmed or refuted to a great extent depending on whether the non-numerical information of the votes used (proportions in the number of votes and their structure) is a suitable indicator of similarity between each pair of users compared. In order to clarify this situation we have applied basic statistical functions to all votes that are usually cast by the users.

Fig. 1 displays the arithmetic average and standard deviation distributions of the votes cast by users of Movielens 1 M and Netflix [5]. As we can see, most of the users' votes are between 3 and 4 stars (arithmetic average), with a variation of approximately 1 star (standard deviation).

By analyzing Fig. 1, we can assume that the 3–4 star interval marks the division between the votes that positively rate the items from those that rate them in a non-positive way. In general, a positive vote will be placed at value 4 and in exceptional cases at 5, whilst a non-positive vote will be placed at value 3 and in exceptional cases at 2 or 1.

Taking into account the very low number of especially negative votes (2 and 1 stars) shown in Graphs 1a and 1b, we can assume that the users have a tendency to not rate items they consider in a non-positive way, whilst to a lesser extent the opposite also occurs: when they cast a vote there is a great probability that it implies a positive rating (above 3.5 stars in Graphs 1a and 1b).

Therefore, it seems that the users tend to simplify their ratings into positive/non-positive and then transfer their psychological choice to the numerical plane. In order to check this hypothesis the following experiment has been carried out [5] on the

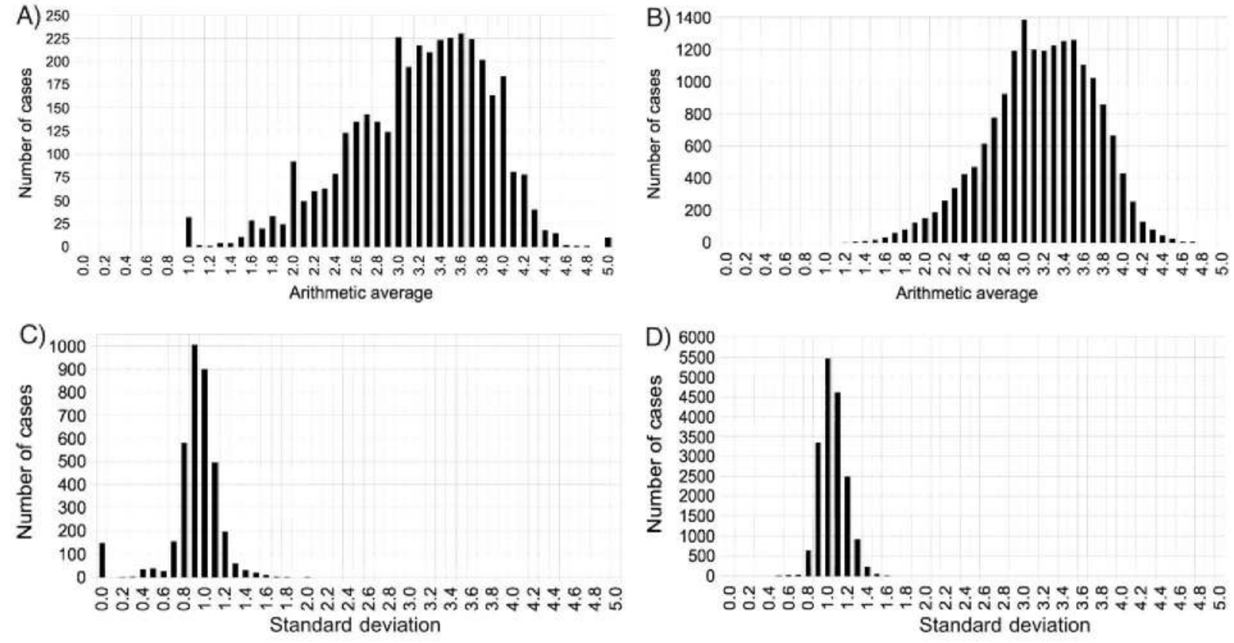

Fig. 1. Arithmetic average and standard deviation on the MovieLens 1 M and NetFlix ratings of the items. (A) Movielens arithmetic average, (B) Netflix arithmetic average, (C) Movielens standard deviation, (D) Netflix standard deviation [5].

Movielens 1 M database: we transformed all 4 and 5 votes into *P* votes (Positive) and all of 1, 2 and 3 votes into *N* votes (Non-positive), in such a way that we aim to measure the impact made on the recommendations by doing without the detailed information provided by the numerical values of the votes.

In the experiment we compare the precision/recall obtained in a regular way (using the numerical values of the votes) with that obtained using only the discretized values *P* and *N*. Fig. 2 displays the results, which show how the "positive/non-positive" discretization not only does not worsen the precision/recall measurements, but rather it improves them both, particularly the precision when the number of recommendations (*N*) is high.

The reasoning shown and the experimental results obtained encourage the use of the non-numerical information of the users' votes as a means of attempting to obtain a cold-start similarity measure which, by making use of this additional information, provides better results than the traditional metrics.

4. Design of the proposed user cold-start similarity measure

The cold-start similarity measure proposed is formed by carrying out a linear combination of a group of simple similarity measures. The scalar values with which each individual similarity measure is weighted are obtained in a process of optimization based on neural learning; this way, after a stage to determine the weights of the linear combination, the cold-start similarity measure can be used to obtain the k-neighbors of each cold-start user who request recommendations.

One of the simple similarity measures is Jaccard, which processes the non-numerical information of the votes; the rest base their operation on the simplest information with which two votes can be compared: their difference.

4.1. Formalization

Given an RS with a database of *L* users and *M* items rated in the range [*min..max*], where the absence of ratings will be represented by the symbol •.

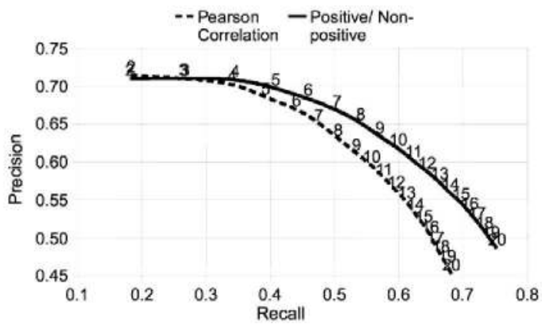

Fig. 2. Precision/Recall obtained by transforming all 4 and 5 votes into *P* votes (Positive) and all 1, 2 and 3 votes into *N* votes (Non-positive), compared to the results obtained using the numerical values. 20% of test users, 20% of test items, *K* = 150, Pearson correlation, relevant threshold = 4. MovieLens 1 M (Bobadilla et al., 2010).

Table 1
Parameters.

Name	Parameters descriptions
L	# Users
M	# Items
min	# Min rating value
max	# Max rating value
k	# Neighborhoods
N	# Recommendations
θ	Recommendation threshold

$U = \{u \in NaturalNumber | u \in \{1..L\}\}$, set of users (1)

$I = \{i \in NaturalNumber | i \in \{1..M\}\}$, set of items (2)

$V = \{v \in NaturalNumber | \min \leqslant v \leqslant \max\} \cup \{\bullet\}$, set of possible votes (3)

$R_u = \{(i, v) | i \in I, v \in V\}$, ratings of user u (4)

We define vote v of user u on item i as $r_{u,i} = v$ (5)

We define the average of the valid votes of user u as $\bar{r}_u$ (6)

We define the cardinality of a set C as its number of valid elements

$$\#C = \#\{x \in C | x \neq \bullet\} \tag{7}$$

In this way:

$$\#R_u = \#\{i \in I | r_{u,i} \neq \bullet\} \tag{8}$$

Below we present the tables of parameters (Table 1), measures (Table 2) and sets (Table 3) used in the formalizations made in the paper.

Table 2
Measures.

Name	Measures descriptions
$v^0_{x,y}$	# Items with the same value in user x and user y (normalized)
$v^1_{x,y}$	# Items with a difference of 1 stars in user x and user y (normalized)
$v^3_{x,y}$	# Items with a difference of 3 stars in user x and user y (normalized)
$v^4_{x,y}$	# Items with a difference of 4 stars in user x and user y (normalized)
$\mu_{x,y}$	Mean squared differences (user x, user y)
$\sigma_{x,y}$	Standard deviation
$Jaccard_{x,y}$	Jaccard similarity measure
$\{w_1,\ldots,w_6\}$	Similarity measure weights
$p_{u,i}$	Prediction to the user on the item
$m_{u,i}$	Prediction error on user u, item i
m_u	User u mean absolute error
m	RS mean absolute error
$c_{u,i}$	User u, item i, coverage
c_u	User u coverage
c	RS coverage
$q_{u,i}$	Is i recommended to the user u?
$t_{u,i}$	Is i recommended relevant to the user u?
t_u	Precision of the user u
t	Precision of the RS
$n_{u,i}$	Is i not recommended relevant to the user u?
x_u	Recall of the user u
x	Recall of the RS

Table 3
Sets.

Name	Sets descriptions	Parameters
U	Users	L
I	Items	M
V	Rating values	*min, max*
R_u	User ratings	*user*
$V^d_{x,y}$	Items rated with a difference of d stars	*user x, user y, d*
$G_{x,y}$	Items rated simultaneously by users x and y	*user x, user y*
R^*_u	Items voted by user u	*user*
K_u	Neighborhoods of the user	*user, k*
P_u	Predictions to the user	*user, k*
$H_{u,i}$	User's neighborhoods which have rated the item	*user, i, k*
X_u	Top recommended items to the user	*user, k, θ*
Z_u	Top N recommended items to the user	*user, k, N, θ*
U^t	Training users	
U^v	Validation users	
I^t	Training items	
I^v	Validation items	
M_u	Items rated by user u where a prediction can be determined	*user*
O	Validation users with assigned MAE	
C_u	Items not rated by user u where a prediction can be determined	*user*
O^*	Validation users with assigned coverage value	
S	Validation users with assigned precision value	
Y_u	Set of recommended relevant items (true-positives)	*user*
N_u	set of not recommended relevant items	*user*
S^*	Validation users with assigned recall value	

4.2. Set of basic similarity measures

In order to find the similitude between two users x and y, we first take all of the information regarding the value of the vote (or lack of vote) from these users in each of the items of the RS. We will assess the following basic similarity measures between two users x and y:

- Measures based on the numerical values of the votes
  1. v^0, v^1, v^2, v^3, v^4 (Eqs. (9)–(11)).
  2. Mean squared differences (μ), (Eqs. (12) and (13)).
  3. Standard deviation of the squared differences (σ), (Eqs. (12) and (14)).
- Measure based on the arrangement of the votes
  4. Jaccard, (Eqs. (15) and (16)).

v^0 represents the number of cases in which the two users have voted with exactly the same score, indicating a high degree of similarity between them; it contributes to increasing the number of cases of particularly accurate predictions.

v^4 represents the opposite case: the number of times that they have voted in a completely opposite way; the aim is to minimize the importance given to the fact that they have voted for the same item, as they have done so by indicating very different preferences. It also contributes to reducing the number of cases of particularly incorrect predictions.

v^1, v^2 and v^3 represents the intermediate cases (v^1 the number of cases in which users have voted with a difference of one score, v^2 with a difference of two scores,...).

μ provides the simplest and most intuitive measure of similitude between users, but it could be a good idea to complement it with the importance held by the extreme cases in this measure, using σ.

The Jaccard measure rewards situations in which the two users have voted for similar sets of items, taking into account the proportion regarding the total number of voted items by both.

$$\text{Let } V^d_{x,y} = \{i \in I | r_{x,i} \neq \bullet \wedge r_{y,i} \neq \bullet \wedge |r_{x,i} - r_{y,i}| = d \text{ where } d \in \{0,\ldots,\max - \min\}\} \tag{9}$$

$$\text{We define } \forall d \in \{0,\ldots,\max - \min\},\ b^d_{x,y} = \#V^d_{x,y} \tag{10}$$

finally, we perform the normalization:

$$v^d_{x,y} = \frac{b^d_{x,y}}{\sum_{d=0}^{\max-\min} b^d_{x,y}}, \quad v^d_{x,y} \in [0,1] \tag{11}$$

Let $G_{x,y} = \{i \in I | r_{x,i} \neq \bullet \wedge r_{y,i} \neq \bullet\}$, the set of items voted simultaneously by both users (12)

$$\mu_{x,y} = 1 - \frac{1}{\#G_{x,y}} \sum_{i \in G_{x,y}} \left(\frac{r_{x,i} - r_{y,i}}{\max - \min}\right)^2 \iff G_{x,y} \neq \emptyset,\ \mu_{x,y} \in [0,1] \tag{13}$$

$$\sigma_{x,y} = \sqrt{\frac{1}{\#G_{x,y}} \sum_{i \in G_{x,y}} \left(\left(\frac{r_{x,i} - r_{y,i}}{\max - \min}\right)^2 - (1 - \mu_{x,y})\right)^2} \iff G_{x,y} \neq \emptyset,\ \sigma_{x,y} \in [0,1] \tag{14}$$

Lets $R^*_u = \{i \in I | r_{u,i} \neq \bullet\}$, the set of items voted by user u (15)

$$Jaccard_{x,y} = \frac{\#(R^*_x \cap R^*_y)}{\#(R^*_x \cup R^*_y)}, \quad Jaccard_{x,y} \in [0,1] \tag{16}$$

4.3. Similarity measures selected

The total set of basic similarity measures to be applied to the proposed metric is as follows:

$$\{v^0, v^1, v^2, v^3, v^4, \sigma, \mu, Jaccard\} \tag{17}$$

With the aim of reducing the number of basic similarity measures, using Netflix and Movielens we have calculated various quality results (MAE, coverage, precision and recall) using the full set of basic similarity measures; subsequently, we repeated the process eight times, eliminating one of the eight basic similarity measures in each repetition (and conserving the other seven). After concluding these experiments, we have rejected the similarity measures which on elimination caused a very slight worsening in the quality results.

Finally, we have seen that the quality results offered by all of the similarity measures selected are only slightly worse than the original ones, and therefore, the significant information is provided by the set of similarity measures which have not been rejected. The empirical results have led us to select the following subset of (17):

$$\{v^0, v^1, v^3, v^4, \mu, Jaccard\} \tag{18}$$

4.4. Formulating the metric

The MJD proposed metric (Mean-Jaccard-Differences) is based on the hypothesis that by combining the six individual similarity measures presented in (18), we will be able to obtain a global similarity measure between pairs of users. As each individual similarity measure presents its relative degrees of importance, it is necessary to assign a weighting (w_i) to each of them. This way, the proposed metric is formulated as:

$$MJD_{x,y} = \frac{1}{6}\left(w_1 v^0_{x,y} + w_2 v^1_{x,y} + w_3 v^3_{x,y} + w_4 v^4_{x,y} + w_5 \mu_{x,y} + w_6 Jaccard_{x,y}\right) \tag{19}$$

At this point, it is necessary to determine the weights w_i which enable us to obtain a metric that improves the results of those commonly used; for this purpose we look for a potential solution to an optimization problem based on neural learning.

As an illustration, using Netflix database, the final weights after the adjustment are those shown in Table 4. The values obtained indicate the importance of each individual measure metric in the final result of the NN measure metric. Note how $w_4 v^4$ (number of cases in which the votes of the two users are totally different) has a very negative impact on the similarity result between the users considered.

By analyzing the weights (w_i) obtained in the neural learning process, we can determine that the proposed similarity measure mainly uses the votes' numerical information, and that this information is complemented and modulated by the arrangement of the votes provided by Jaccard. The numerical information is based on the measure of μ; furthermore, the similarity between users is reinforced with the results of v^0 and v^1 and is reduced with the results of v^3 and v^4.

4.5. Neural network learning

Eq. (19) has a very similar form to the input (NET) of an artificial neural network, and more specifically to an ADALINE network. As it is a problem of adjustment, and not of classification, we can use a continuous activation function, e.g. linear activation, instead of the sigmoid function used in the traditional perceptron. A system of this sort would technically be a perceptron with linear activation function. This parallelism between our metric and the propagation of the signal in a perceptron enables the Widrow–Hoff method [54] to be used, adapted to our problem, to make the adjustment of the weights using the gradient descent method:

$$w_i(t+1) = w_i(t) + \alpha \cdot x_i(MAE(t)_{MJD(t)} - MAE_{JMSD}) \quad i \in \{1,\ldots,6\}$$
$$\text{where } x_1 = v^0_{u1,u2},\ x_2 = v^1_{u1,u2},\ x_3 = v^3_{u1,u2},\ x_4 = v^4_{u1,u2},$$
$$x_5 = \mu_{u1,u2},\ x_6 = Jaccard_{u1,u2} \tag{20}$$

The learning of the neuronal network is carried out using a set of pairs of users (u_1,u_2) where u_1 represents a cold-start user (who has rated between 2 and 20 items) and u_2 represents any user in the database. This pair of users are taken into account for updating the weights w_i.

$MAE(t)_{MJD(t)}$ stands for the MAE of the recommender system, which is calculated using the proposed metric MJD based on the set of weights $w_i(t)$. This measure considers that only the cold-start users are test users, and measures the accuracy of the recommender system based on the MJD in the instant t.

MAE_{JMSD} stands for the MAE of the recommender system calculated taking into account all test users (not only cold-start users) and the similarity measure JMSD. This measure represents the accuracy of the recommender system which we try to reach.

The measure MAE_{JMSD} is an upper bound of the accuracy obtained in each instant t ($MAE(t)_{MJD(t)}$) since we can reach better accuracy using all users as test users than using only cold-start users. The metric JMSD has been selected as a reference since it provides good results [5].

For the adjustment of the weights it is necessary to develop a training set in which the following are specified:

- The input data to the system for every pair of users, in our case the values presented in (18).
- The desired output for each pair of users of the system. To implement the error measure, we use the system MAE_{JMSD}, making use of a very small set of test users with the aim of achieving reasonable execution times. These calculations are obtained using parallel processing through a cluster of computers.

Fig. 3 shows the main modules involved in the whole neural learning process, where the time $t+1$ weights are adjusted in accordance with the $MAE(t)_{MJD(t)}$ obtained.

Table 4
Weights obtained using the gradient descent method (Netflix RS).

w_1	w_2	w_3	w_4	w_5	w_6
0.66	0.35	−0.21	−0.43	1.07	0.31

5. Collaborative filtering specifications

In this section we specify the CF methods proposed to make recommendations. We also formalize the CF methodology used in the experiments; through this methodology, we calculate the quality results of the predictions and recommendations for the similarity measures studied.

Due to the scarce number of items voted for by the cold-start users (which we have determined in the interval {2,…,20}), we have decided to use leave-one-out cross validation to ensure the greatest possible number of training items in each validation process. The proposed methodology includes the formalization of the processes to obtain the MAE, coverage, precision and recall using leave-one-out cross validation.

5.1. Obtaining prediction and recommendations

5.1.1. Users's k-neighbors

We define K_u as the set of k neighbors of the user u and we use the desired user similarity measure: $\text{sim}_{x,y}$, where x and y are users. The following must hold:

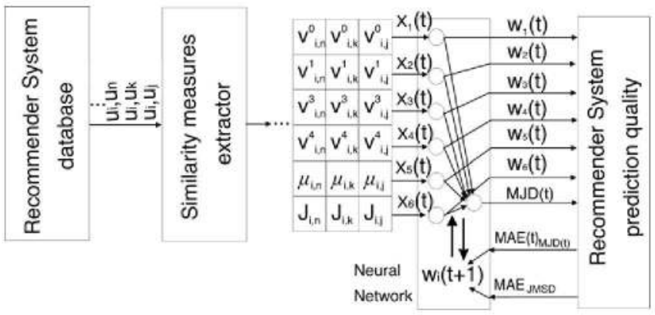
Recommender System database
ui,un
ui,uk
ui,uj
Similarity measures extractor
$V^0_{i,n}$ $V^0_{i,k}$ $V^0_{i,j}$
$V^1_{i,n}$ $V^1_{i,k}$ $V^1_{i,j}$
$V^3_{i,n}$ $V^3_{i,k}$ $V^3_{i,j}$
$V^4_{i,n}$ $V^4_{i,k}$ $V^4_{i,j}$
$\mu_{i,n}$ $\mu_{i,k}$ $\mu_{i,j}$
$J_{i,n}$ $J_{i,k}$ $J_{i,j}$
X1(t)
X2(t)
X3(t)
X4(t)
X5(t)
X6(t)
w1(t)
w2(t)
w3(t)
w4(t)
w5(t)
w6(t)
MJD(t)
MAE(t)MJD(t)
wi(t+1)
MAE JMSD
Neural Network
Recommender System prediction quality

Fig. 3. Neural learning process.

$$K_u \subset U \wedge \ \#K_u = k \wedge \ u \notin K_u \tag{21}$$

$$\forall x \in K_u, \forall y \in (U - K_u), sim_{u,x} \geqslant sim_{u,y} \tag{22}$$

5.1.2. Prediction of the value of an item

Based on the information provided by the k-neighbors of a user u, the CF process enables the value of an item to be predicted as follows:

$$\text{Let } P_u = \{(i,p) | i \in I, \ p \in \text{RealNumber}\}, \\ \text{set of prediction to the user } u \tag{23}$$

$$\text{We will assign the value of the prediction } p \\ \text{made to user } u \text{ on item } i \text{ as } p_{u,i} = p \tag{24}$$

Once the set of k users (neighbors) similar to active u has been calculated (K_u), in order to obtain the prediction of item i on user u (24), we use the aggregation approach deviation-from-mean (Eqs. (25)–(27)).

$$\text{Lets } H_{u,i} = \{n \in K_u | r_{n,i} \neq \bullet\} \tag{25}$$

$$p_{u,i} = \bar{r}_u + \frac{1}{\sum_{n \in H_{u,i}} sim_{u,n}} \sum_{n \in H_{u,i}} sim_{u,n}(r_{n,i} - \bar{r}_n) \Longleftrightarrow H_{u,i} \neq \emptyset \tag{26}$$

$$p_{u,i} = \bullet \Longleftrightarrow H_{u,i} = \emptyset \tag{27}$$

5.1.3. Top N recommendations

We define X_u as the set of predictions to user u, and Z_u as the set of N recommendations to user u.

The following must hold:

$$X_u \subset I \ \forall i \in X_u, \quad r_{u,i} = \bullet, \quad p_{u,i} \neq \bullet, \tag{28}$$

$$Z_u \subseteq X_u, \#Z_u = N, \quad \forall x \in Z_u, \quad \forall y \in X_u \ p_{u,x} \geqslant p_{u,y} \tag{29}$$

If we want to impose a minimum recommendation value: $\theta \in$ RealNumber, we add $p_{u,i} \geqslant \theta$

5.1.4. Running example

We define a micro RS example with 6 users, 12 items and a range of votes from 1 to 5:

$$\mathrm{U} = \{\mathrm{u}_1, \mathrm{u}_2, \mathrm{u}_3, \mathrm{u}_4, \mathrm{u}_5, \mathrm{u}_6\},$$
$$\mathrm{I} = \{\mathrm{i}_1, \mathrm{i}_2, \mathrm{i}_3, \mathrm{i}_4, \mathrm{i}_5, \mathrm{i}_6, \mathrm{i}_7, \mathrm{i}_8, \mathrm{i}_9, \mathrm{i}_{10}, \mathrm{i}_{11}, \mathrm{i}_{12}\}, \quad \mathrm{V} = \{1,2,3,4,5,\bullet\}$$

Table 5 shows the votes ($r_{u,i}$) cast by each user.

In order to make recommendations for user 1 (by way of example), using MJD and the weights of Table 4, we calculate their similarity with the rest of the users (Table 6).

Taking $k = 2$ neighbors, we obtain the set of neighbors of $u_1: K_{u_1} = \{u_5, u_4\}$.

To calculate the possible predictions that can be made to u_1, we use the arithmetic average as an aggregation approach instead of the equation proposed in (27). The predictions obtained are summarized in Table 7.

Table 5
RS running example (users, items and ratings).

$r_{u,i}$	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1	•	2	•	•	3	5	•	•	3	•	•	•
u_2	1	•	•	2	5	•	•	•	4	•	5	•
u_3	•	•	3	•	•	4	•	5	1	•	2	5
u_4	2	3	•	•	4	•	5	4	•	2	3	1
u_5	•	3	4	3	•	5	4	•	2	2	3	4
u_6	5	4	2	3	•	3	2	3	5	3	•	3

Table 6
Similarity measures between users using MJD.

MJD	u_2	u_3	u_4	u_5	u_6
u_1	1.166	1.155	1.415	1.572	0.887

Table 7
Predictions that u_1 can receive using MJD, $k = 2$ and arithmetic average as aggregation approach.

$P_{u1,i}$	i_1	i_3	i_4	i_7	i_8	i_{10}	i_{11}	i_{12}
u_1	2	4	3	4,5	4	2	3	2,5

If we want to make $N = 3$ recommendations: $Z_u = \{i_7, i_3, i_8\}$.

5.2. Obtaining the collaborative filtering quality measures

In this section we specify the way in which we will obtain the results of the quality offered by the proposed similarity measure. We provide equations which formalize the process of obtaining the selected prediction quality measures: MAE and coverage, and the selected recommendation quality measures: precision and recall.

We have determined that a user will be considered as a cold-start user when they have between 2 and 20 items voted. This very limited number of items involves that the use of the most common cross validation (random sub-sampling and k-fold cross validation) is not appropriated either in the prediction quality measures or in the recommendation quality measures; there are not enough items to feed suitably the training and validation stages.

The cross validation method chosen to carry out the experiments is leave-one-out cross validation; this method involves using a single observation from the original sample as the validation data, and the remaining observations as the training data. Each observation in the sample is used once as the validation data. This method is computationally expensive, but it allows us to use larger sets of training data.

Starting from the sets of users and items defined in RS (Eqs. (1) and (2)), we establish the sets of training and validation users and training and validation items which will be used in each individual validation process of an item using leave-one-out cross validation.

$$U^t \subset U, \text{ set of training users} \tag{30}$$

$$U^v \subset U, \text{ set of validation users} \tag{31}$$

$$I^t \subset I, \text{ set of training items} \tag{32}$$

$$I^v \subset I, \text{ set containing the validation item} \tag{33}$$

Using leave-one-out cross validation, the following holds:

$$U^v \cup U^t = U, \quad U^v \cap U^t = \emptyset, \quad \#I^v = 1, \quad I^v \cup I^t = I, \quad I^v \cap I^t = \emptyset \tag{34}$$

In the process to obtain the RS quality measures we use validation items, reserving the training items to determine the k-neighbors. Therefore, the similarity measures make their calculations using training items; Eq. (35) replaces Eq. (12). In the same way, Eq. (22) must reflect that the predictions are made on the validation items (Eq. (36)).

$$G_{x,y} = \{i \in I^t | r_{x,i} \neq \bullet \wedge r_{y,i} \neq \bullet\} \tag{35}$$

$$P_u = \{(i,p) | i \in I^v,\ p \in \text{RealNumber}\} | u \in U^v \tag{36}$$

We modify the equations to obtain k-neighborhoods (21) and (22), obtaining Eqs. (37) and (38).

$$u \in U^v\ K_u \subset U^t \quad \#K_u = k \quad u \notin K_u \tag{37}$$

$$\forall x \in K_u,\ \forall y \in (U^t - K_u),\ sim_{u,x} \geqslant sim_{u,y} \tag{38}$$

5.2.1. Quality of the prediction: mean absolute error/accuracy

In order to measure the accuracy of the results of a RS, it is usual to use the calculation of some of the most common error metrics, amongst which the mean absolute error (MAE) and its related metrics: mean squared error, root mean squared error, and normalized mean absolute error stand out. The MAE indicates the average error made in the predictions; therefore, the lower this value, the better the accuracy of the system.

Using leave-one-out cross validation, we carry out a validation process for each item of each validation user:

$$\text{Let } u \in U^v \wedge i \in I | \#R_u \in \{2,\dots,20\} \wedge r_{u,i} \neq \bullet \tag{39}$$

$$I^v = \{i\},\ I^t = \{j \in I | r_{u,i} \neq \bullet \wedge j \neq i\} \tag{40}$$

Through I^t we can calculate the k-neighbors (K_u). Through K_u and I^v, the prediction $p_{u,i}$ can be calculated.

We define the absolute error of a user u on an item $i(m_{u,i})$ as:

$$m_{u,i} = |p_{u,i} - r_{u,i}| \iff u \in U^v,\ i \in I^v,\ p_{u,i} \neq \bullet \wedge r_{u,i} \neq \bullet,\quad m_{u,i} \in [0, \max - \min] \tag{41}$$

$$m_{u,i} = \bullet \iff u \in U^v,\ i \in I^v,\ p_{u,i} = \bullet \vee r_{u,i} = \bullet \tag{42}$$

The MAE of the user $u(m_u)$ is obtained as the average of its $m_{u,i}$:

$$\text{Let } M_u = \{i \in I | m_{u,i} \neq \bullet\},\ u \in U^v \tag{43}$$

$$m_u = \frac{1}{\#M_u} \sum_{i \in M_u} m_{u,i}, \quad m_u \in [0, \max - \min] \tag{44}$$

The MAE of the RS: (m) is obtained as the average of the user's MAE:

$$\text{Let } O = \{u \in U^v | m_u \neq \bullet\} \tag{45}$$

We define the system's MAE as:

$$m = \frac{1}{\#O} \sum_{u \in O} m_u \iff O \neq \emptyset, \quad m \in [0, \max - \min] \tag{46}$$

$$m = \bullet \iff O = \emptyset \tag{47}$$

The accuracy is defined as:

$$accuracy = 1 - \frac{m}{\max - \min}, \quad accuracy \in [0,1] \tag{48}$$

5.2.2. Quality of the prediction: coverage

The coverage could be defined as the capacity of predicting from a metric applied to a specific RS. In short, it calculates the percentage of situations in which at least one k-neighbors of each active user can rate an item that has not been rated by that active user. Once again, using leave-one-out cross validation we carry out a validation process for each item of each validation user:

Lets $u \in U^v \wedge i \in I | \#R_u \in \{2,\dots,20\} \wedge r_{u,i} \neq \bullet$

$$I^v = \{i\},\ I^t = \{j \in I | r_{u,i} \neq \bullet \wedge j \neq i\} \tag{49}$$

With I^t we obtain k-neighbors (K_u) and with K_u and I^v we make the prediction $p_{u,i}$.

We define the coverage of a user u on an item i as $c_{u,i}$; this value indicates that we can make a prediction of item i to user u.

$$\left.\begin{array}{l} c_{u,i} \neq \bullet \iff u \in U^v,\ i \in I^v,\ p_{u,i} \neq \bullet \wedge r_{u,i} = \bullet \\ c_{u,i} = \bullet \iff u \in U^v,\ i \in I^v,\ p_{u,i} = \bullet \vee r_{u,i} \neq \bullet \end{array}\right\} \tag{50}$$

Let C_u be the set of items on which a prediction can be made to user u:

$$C_u = \{i \in I | c_{u,i} \neq \bullet \quad where\ u \in U^v\} \tag{51}$$

The coverage of the user u (c_u) is obtained as the proportion between the number of items not voted for by the user which can be predicted and the total items not voted for by the user.

$$c_u = 100 \times \frac{\#C_u}{\#I - \#R_u} \iff R_u \neq I, \quad c_u \in [0, 100] \tag{52}$$

$$c_u = \bullet \iff R_u = I \tag{53}$$

The coverage of the RS: (c) is obtained as the average of the user's coverage:

$$\text{Lets } O^* = \{u \in U^v | c_u \neq \bullet\} \tag{54}$$

We define the system's coverage as:

$$c = \frac{1}{\#O^*} \sum_{u \in O^*} c_u \iff O^* \neq \emptyset, \quad c \in [0, 100] \tag{55}$$

$$c = \bullet \iff O^* = \emptyset \tag{56}$$

5.2.3. Quality of the recommendation: precision

The precision refers to the capacity to obtain relevant recommendations regarding the total number of recommendations made. We define θ as the minimum value of a vote to be considered relevant.

Let $u \in U^v \wedge i \in I | \#R_u \in \{2,\dots,20\} \wedge r_{u,i} \neq \bullet$

$$I^v = \{i\},\ I^t = \{j \in I | r_{u,i} \neq \bullet \wedge j \neq i\} \tag{57}$$

Through I^t we obtain k-neighbors (K_u), and through K_u and I^v we make the prediction $p_{u,i}$.

Each $q_{u,i}$ term indicates whether item i has been recommended to user u.

$$\left.\begin{array}{ll} q_{u,i} \neq \bullet \iff u \in U^v, i \in I^v \quad (p_{u,i} \neq \bullet \wedge p_{u,i} \geqslant \theta) \wedge r_{u,i} \neq \bullet, & \text{recommended item} \\ q_{u,i} = \bullet \iff u \in U^v,\ i \in I^v \quad (p_{u,i} = \bullet \vee p_{u,i} < \theta) \wedge r_{u,i} \neq \bullet, & \text{not recommended item} \end{array}\right\} \tag{58}$$

Each $t_{u,i}$ term indicates whether item i recommended to user u has been relevant.

$$\left.\begin{array}{ll} t_{u,i} \neq \bullet \iff u \in U^v, i \in I^v \quad q_{u,i} \neq \bullet \wedge r_{u,i} \geqslant \theta, & \text{recommended and relevant} \\ t_{u,i} = \bullet \iff u \in U^v,\ i \in I^v \quad q_{u,i} \neq \bullet \wedge r_{u,i} < \theta, & \text{recommended and not relevant} \end{array}\right\} \tag{59}$$

The precision of the user $u(t_u)$ is obtained as the proportion between the number of recommended relevant items to the user and the total items recommended to the user.

$$t_u = \frac{\#\{i \in I | t_{u,i} \neq \bullet\}}{\#\{i \in I | q_{u,i} \neq \bullet\}} \iff \{i \in I | q_{u,i} \neq \bullet\} \neq \emptyset, \quad t_u \in [0,1] \tag{60}$$

$$t_u = \bullet \iff \{i \in I | q_{u,i} \neq \bullet\} = \emptyset \tag{61}$$

The precision of the RS: (t) is obtained as the average of the user's precision:

$$\text{Let } S = \{u \in U^v | t_u \neq \bullet\} \tag{62}$$

We define the system's precision as:

$$t = \frac{1}{\#S} \sum_{u \in S} t_u \Longleftrightarrow S \neq \emptyset, \quad t \in [0,1] \tag{63}$$

$$t = \bullet \Longleftrightarrow S = \emptyset \tag{64}$$

5.2.4. Quality of the recommendation: recall

The recall refers to the capacity to obtain relevant recommendations regarding the total number of relevant items.

We define θ as the minimum value of a vote to be considered as relevant.

Lets $u \in U^v \wedge i \in I | \#R_u \in \{2, \ldots, 20\} \wedge r_{u,i} \neq \bullet$

$$I^v = \{i\}, \; I^t = \{j \in I | r_{u,i} \neq \bullet \wedge j \neq i\} \tag{65}$$

With I^t we obtain k-neighborhoods (K_u) and with K_u and I^v we make prediction $p_{u,i}$.

Each term $n_{u,i}$ indicates whether item i not recommended to the user u is relevant

$$\left.\begin{array}{l} n_{u,i} \neq \bullet \Longleftrightarrow u \in U^v, \; i \in I^v \quad p_{u,i} \neq \bullet \wedge r_{u,i} \neq \bullet \wedge p_{u,i} < \theta \wedge r_{u,i} \geqslant \theta, \text{ not rec. \&rel.} \\ n_{u,i} = \bullet \Longleftrightarrow u \in U^v, \; i \in I^v \quad p_{u,i} \neq \bullet \wedge r_{u,i} \neq \bullet \wedge p_{u,i} < \theta \wedge r_{u,i} < \theta, \text{ not rec. \&not rel.} \end{array}\right\} \tag{66}$$

We define Yu as set of recommended items to user u which have been relevant.

$$Y_u = \{i \in I | t_{u,i} \neq \bullet\}, \; u \in U^v \tag{67}$$

We define Nu as the set of not recommended items to user u which have been relevant.

$$N_u = \{i \in I | n_{u,i} \neq \bullet\}, \; u \in U^v \tag{68}$$

The recall of the user $u(x_u)$ is obtained as the proportion between the number of recommended relevant items to the user and the total relevant items for the user (recommended and not recommended).

$$x_u = \frac{\#Y_u}{\#(Y_u \cup N_u)} \Longleftrightarrow Y_u \cup N_u \neq \emptyset, \quad x_u \in [0,1] \tag{69}$$

$$x_u = \bullet \Longleftrightarrow Y_u \cup N_u = \emptyset \tag{70}$$

The recall of the RS: (x) is obtained as the average of the user's recall:

$$\text{Lets } S^* = \{u \in U^v | x_u \neq \bullet\} \tag{71}$$

We define the system's recall as:

$$x = \frac{1}{\#S^*} \sum_{u \in S^*} x_u \Longleftrightarrow S^* \neq \emptyset, \quad x \in [0,1] \tag{72}$$

$$x = \bullet \Longleftrightarrow S^* = \emptyset \tag{73}$$

5.2.5. Running example

We establish $U^v = \{u_1, u_2\}$, $U^t = \{u_3, u_4, u_5, u_6\}$, $k = 2$.

Table 8 shows an outline of the process with which we obtain the quality measures MAE, precision and recall of user $u1$. The two first columns (I^v and I^t) describe respectively the validation and training items. The third column (MJD$_{u1,ui}$) informs about the similarity between $u1$ and each validation user. The fourth column (K_{u1}) describes the k-neighbors of $u1$. The fifth column ($P_{u1,Iv}$) shows the prediction of item i on user $u1$ using the arithmetic mean as aggregation approach instead of Eq. (27). The sixth column ($r_{u1,Iv}$) informs about the vote of U1 for the validation items: the column 'MAE' describes the mean absolute difference between prediction and vote (41); the column 'Precision' indicates when an item has been recommended (q!=) and when it has been recommended and it is relevant (t!=); the column 'Recall' shows when a relevant item has been recommended (t!=), and when it has not (n!=).

In the example, we make the following calculations which determine the similarity between users u_1 and u_3 : $G_{u1,u3} = \{i_6, i_9\}$, $v^0_{u1,u3} = 0/2 = 0$, $v^1_{u1,u3} = 1/2 = 0.5$, $v^3_{u1,u3} = 0/2 = 0$, $v^4_{u1,u3} = 0/2 = 0$, $Jaccard_{u1,u3} = 2/7 = 0.286$,

$$\mu_{u1,u3} = 1 - \frac{\left(\frac{5}{5}\frac{4}{1}\right)^2 + \left(\frac{3}{5}\frac{1}{1}\right)^2}{2} = 0.844$$

$$MJD_{u1,u3} = \frac{1}{6}\left(w_1 v^0_{u1,u3} + w_2 v^1_{u1,u3} + w_3 v^3_{u1,u3} + w_4 v^4_{u1,u3} + w_5 \mu_{u1,u3} + w_6 Jaccard_{u1,u3}\right) = 1.166$$

Table 9 specifies the way to obtain the coverage of u_1 with each of their not voted items. Based on the similarity results set out in Table 6 and the predictions expressed in Table 7 we determine 100% coverage for user u_1.

Table 10 summarizes the results provided by the quality measures applied to the running example, using MJD.

6. Design of the experiments

The experiments have been carried out using the Netflix and Movielens databases, which contains the cold-start users filtered

Table 9
User u_1 coverage.

I^v	I^t	MJD$_{u1,ui}$				K_{u1}	$P_{u1,\{i\}}$	Quality measures coverage
		u_3	u_4	u_5	u_6			
$\{i_1\}$	$\{i_2,i_5,i_6,i_9\}$	1.166	1.415	1.572	0.887	$\{u_5,u_4\}$	2	$c \neq \bullet$
$\{i_3\}$	$\{i_2,i_5,i_6,i_9\}$						4	$c \neq \bullet$
$\{i_4\}$	$\{i_2,i_5,i_6,i_9\}$						3	$c \neq \bullet$
$\{i_7\}$	$\{i_2,i_5,i_6,i_9\}$						4,5	$c \neq \bullet$
$\{i_8\}$	$\{i_2,i_5,i_6,i_9\}$						4	$c \neq \bullet$
$\{i_{10}\}$	$\{i_2,i_5,i_6,i_9\}$						2	$c \neq \bullet$
$\{i_{11}\}$	$\{i_2,i_5,i_6,i_9\}$						3	$c \neq \bullet$
$\{i_{12}\}$	$\{i_2,i_5,i_6,i_9\}$						2,5	$c \neq \bullet$
							Total	100

Table 10
Quality measures results.

	Quality measures			
	MAE	Coverage	Precision	Recall
u_1	0.75	100	0.5	1
u_2	1.7	100	1	0.33
Total	1.225	100	0.75	0.66

Table 8
User u_1 MAE, precision and recall.

I^v	I^t	MJD$_{u1,ui}$				K_{u1}	$P_{u1,Iv}$	$r_{u1,Iv}$	Quality measures		
		u_3	u_4	u_5	u_6				MAE	Precision	Recall
$\{i_2\}$	$\{i_5,i_6,i_9\}$	1.166	1.387	1.610	0.864	$\{u_5,u_4\}$	3	2	1	$q = \bullet, t = \bullet$	$t = \bullet, n = \bullet$
$\{i_5\}$	$\{i_2,i_6,i_9\}$	1.166	1.387	1.582	0.895	$\{u_5,u_4\}$	4	3	1	$q \neq \bullet, t = \bullet$	$t = \bullet, n = \bullet$
$\{i_6\}$	$\{i_2,i_5,i_9\}$	0.846	1.422	1.422	0,864	$\{u_4,u_5\}$	5	5	0	$q \neq \bullet, t \neq \bullet$	$t \neq \bullet, n = \bullet$
$\{i_9\}$	$\{i_2,i_5,i_6\}$	1.397	1.422	1.610	0.864	$\{u_5,u_4\}$	2	3	1	$q = \bullet, t = \bullet$	$t = \bullet, n = \bullet$
							Total		0.75	0.5	1

Table 11
Experiments performed.

Databases		# Neighbors on x axis					
	Test users	K (MAE, coverage)		Precision, recall			
		Range	step	K	N	θ	Figures
Movielens 1 M	20%	{100,...,2000}	100	700	{2,...,20}	5	Fig. 4
Netflix	20%	{100,...,2000}	100	700	{2,...,20}	5	Fig. 6
Databases		# Ratings on x axis					
	Test users	(MAE, coverage, precision, recall) #Ratings		K	N	θ	Figures
Movielens 1 M	20%	{2,...,20} step 1		700	10	5	Fig. 5
Netflix	20%	{2,...,20} step 1		700	10	5	Fig. 7

in other databases (users with less than 20 votes). Movielens is the RS research database reference and Netflix offers us a large database on which metrics, algorithms, programming and systems are put to the test. The main parameters of Netflix are: 480189 users, 17770 items, 100,480,507 ratings, 1–5 stars; the main parameters of Movielens are: 6040 users, 3706 items, 1,480,507 ratings, 1 to 5 stars.

Since the database Movielens does not take into account cold-start users (users with less than 20 votes), we have removed votes of this database in order to achieve cold-start users. Indeed, we have removed randomly between 5 and 20 votes of those users who have rated between 20 and 30 items. In this way, those users who now result to rate between 2 and 20 items are regarded as cold-start users. We recover the removing votes of those users with greater than 20 votes despite of removing some their votes (in this way, these users keep immutable in the database).

With the aim of checking the correct operation of the cold-start similarity measure proposed in the paper, it is compared with part of the traditional similarity measures most commonly used in the field of CF: Pearson correlation, cosine and constrained Pearson correlation; with the new metric JMSD [5] and with current user cold-start metrics working just on the users' ratings matrix: PIP [22] and UError [16]. The quality measures to which all of the metrics will be subjected are MAE, coverage, precision and recall, using leave-one-out cross validation for the items, 20% of validation users, 80% of training users (only those who have voted for a maximum of 20 items are processed into the validation users set). Section 5.2 sets out the formalization which gives a detailed description of how to obtain each quality measure result.

Each quality measure (MAE, coverage, precision and recall) will be calculated, using Movielens and Netflix, in two experiments:

Experiment 1. Evolution of the results of MJD, PIP, UError, correlation, constrained correlation, cosine and JMSD. Experiment 1.1: MAE and coverage throughout the range of neighborhoods $k \in \{100,\ldots,2000\}$, step 100. Experiment 1.2: precision and recall throughout the range of number of recommendations $N \in \{2,\ldots,20\}$. In this experiment we make use of all the cold-start users (no more than 20 votes) belonging to the users validation set (U^v).

Experiment 2. Evolution of the results of MJD, PIP, UError, correlation, constrained correlation, cosine and JMSD throughout the range of votes cast by the cold-start users $\#R_u \in \{2,\ldots,20\}$. We use the fixed value of k-neighbors $k = 700$ and number of recommendations $N = 10$.

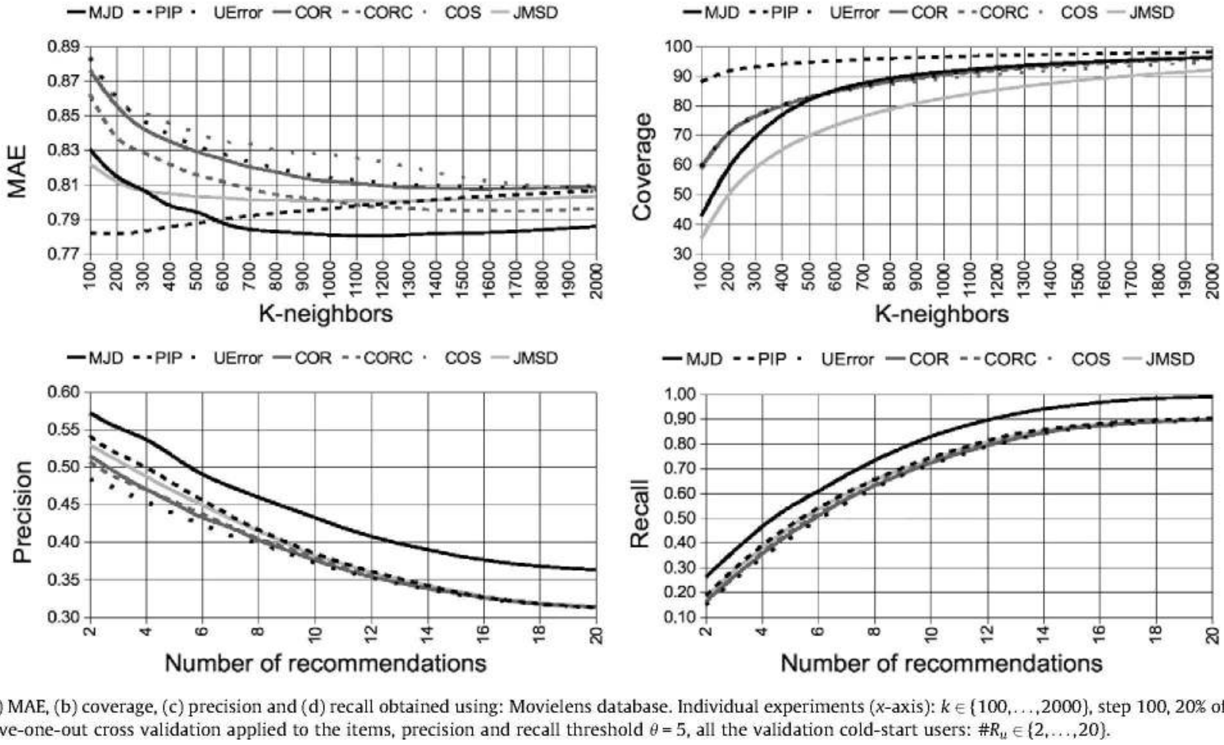

Fig. 4. (a) MAE, (b) coverage, (c) precision and (d) recall obtained using: Movielens database. Individual experiments (x-axis): $k \in \{100,\ldots,2000\}$, step 100, 20% of validation users, leave-one-out cross validation applied to the items, precision and recall threshold $\theta = 5$, all the validation cold-start users: $\#R_u \in \{2,\ldots,20\}$.

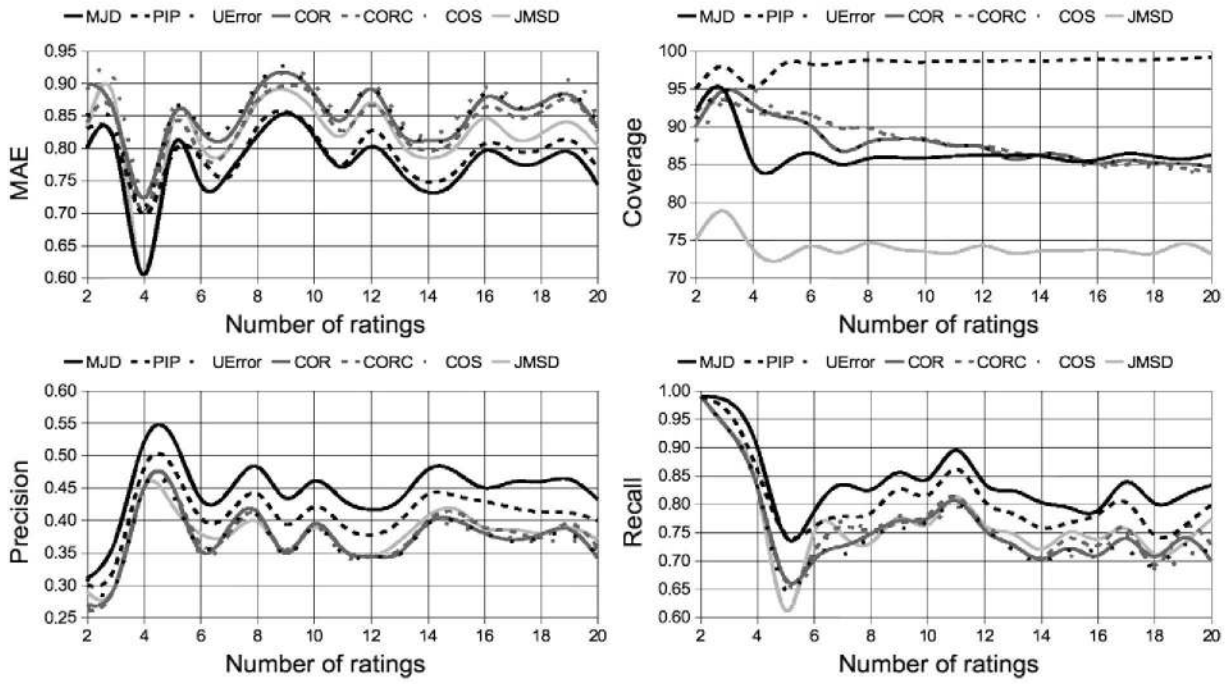

Fig. 5. (a) MAE, (b) coverage, (c) precision and (d) recall obtained using: Movielens database. Individual experiments (x-axis): $\#R_u \in \{2,\ldots,20\}$, 20% of validation users, leave-one-out cross validation applied to the items, precision and recall threshold $\theta = 5$, $k = 700$, $N = 10$.

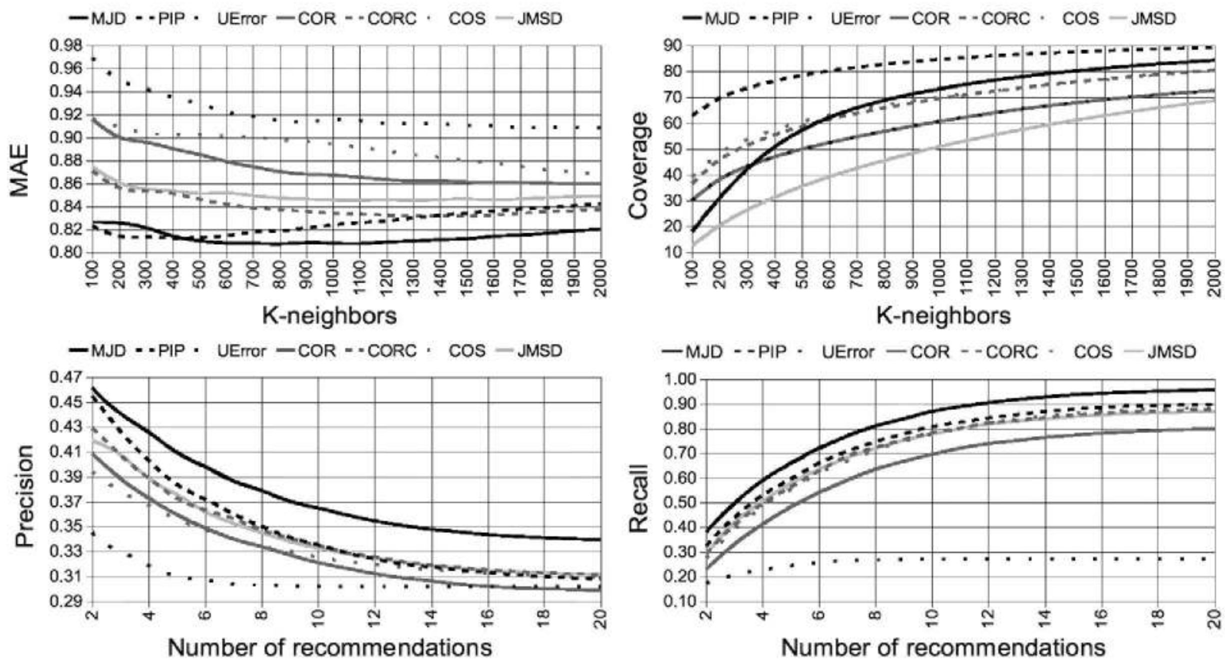

Fig. 6. (a) MAE, (b) coverage, (c) precision and (d) recall obtained using: Netflix database. Individual experiments (x-axis): $k \in \{100,\ldots,2000\}$, step 100, 20% of validation users, leave-one-out cross validation applied to the items, precision and recall threshold $\theta = 5$, all the validation cold-start users: $\#R_u \in \{2,\ldots,20\}$.

Table 11 summarizes the experiments performed (used databases, selected parameters values and figures where the results are shown).

7. Results

Fig. 4 shows the results corresponding to Experiment 1 using Movielens. Graph 1a confirms the paper's hypothesis in the sense that the similarity measure designed (MJD) improves the prediction quality of the traditional similarity measures when they are applied to cold-start users as well as it improves the new user cold-start error-reflected (UError) metric. The PIP metric works better only for a number of neighbors fewer than 500.

Fig. 4(b) displays the negative aspect of the similarity measure proposed which had been highlighted in section 2 (motivation): by selecting neighbors who have a similar number of votes to the active user (using Jaccard), the coverage is weakened. As we can see

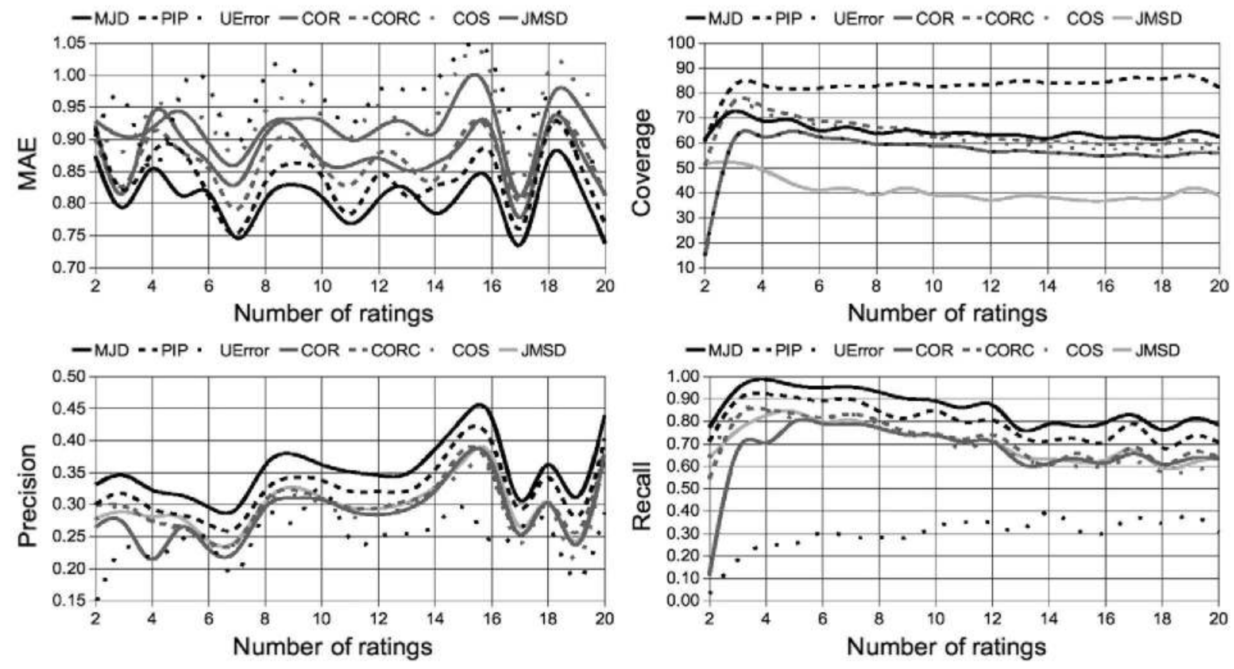

Fig. 7. (a) MAE, (b) coverage, (c) precision and (d) recall obtained using: Netflix database. Individual experiments (x-axis): $\#R_u \in \{2,\ldots,20\}$, 20% of validation users, leave-one-out cross validation applied to the items, precision and recall threshold $\theta = 5$, $k = 700$, $N = 10$.

in Graph 4b, the MJD coverage is worse than the obtained using the other metrics (except JMSD). Graphs 4a and 4b, together, enable the administrator of the RS to make a decision based on the number of neighbors (k) to be used depending on the desired balance between the quality and the variety of the predictions that we wish to offer the cold-start users.

The quality of the recommendations, measured with precision quality measure (proportion of relevant recommendations as regards the total number of recommendations made), improves using the similarity measure proposed (MJD) as regards traditional and cold-start similarity measures (Fig. 4(c)). The improvement is obtained through the whole range of neighbors, which means that the improvement achieved in the predictions is transferred to the recommendations (using $k = 700$).

By analyzing Graph 4d we can determine that the quality of the recommendations measured with the recall quality measure (proportion of relevant recommendations as regards the total number of relevant items) improves, using MJD, for the entire number of neighbors considered.

Fig. 5 (Experiment 2) enables us to discover the quality of the predictions and the recommendations made to the cold-start users according to the number of items they have voted for. In Fig. 5(a) we can see a generalized improvement in the accuracy obtained using the similarity measure proposed (MJD) as regards the traditional and the cold-start ones.

As is to be expected, the cold-start users who have voted for very few items (two or three items) generate greater prediction errors. These cold-start users, which we could call extreme cold-start users, do not present significant improvements in the MAE using MJD; the basic problem with these users is that, in their case, the number of items with which the similarity measures can work is so small that the improvement margin is practically zero. The rest of the cold-start users considered present a reduction in the MAE using MJD as regards the other similarity measures used.

Fig. 5(b) shows that MJD predicts worse than most metrics (especially PIP); using MJD, the parameter that determines the adjustment in the coverage is k (number of neighbors).

Fig. 5(c) shows us precision values obtained with MJD which, as in the MAE, are moderate for the extreme cold-start users and better for the rest of the cold-start users. Fig. 4(d) shows recall positive margins of improvement similar to those obtained when dealing with precision measures; this indicates a good capacity of MJD to reduce the number of false-negatives (not recommended relevant items). Both measures show an improvement in the proposed metric.

Fig. 6 shows the results corresponding to Experiment 1 using Netflix. As may be seen, these results confirm the conclusions derived from the results obtained from the database Movilens. In Fig. 6(a) we can see how MJD provides much better results in relation to PIP when working with Netflix than when working with Movielens. As may be seen in Fig. 4(b) and Fig. 6(b), the coverage results obtained for Netflix are very similar to the ones obtained for Movielens. Figs. 6(c) and (d) show outstanding results in the recommendation quality measures for the database Netflix in analogous way as those obtained for the database Movielens (Figs. 4(c) and (d)).

Fig. 7 (Experiment 2) shows the excellent general behavior of the proposed metric for the database Netflix, in a similar way to the results we obtained for Movielens.

Besides, we have compared the time required to process for the proposed metric MJD and for the metric PIP. Since the calculation of our metric is very simple, it provides much faster recommendations. Using Movielens as recommender system, we have taken all of the cold-start users and we have calculated their similarity with the rest of users in the database. We have repeated this experiment 100 times and we have obtained that the process time of a cold-start user for the MJD similarity metric is 9.11 ms, while for the PIP similarity metric is 66.42 ms (which means a performance improvement of 729%).

8. Conclusions

The new user cold start issue represents a serious problem in RS as it can lead to loss of new users due to the lack of accuracy in their recommendations because of having not made enough votes in the RS. For this reason, it is particularly important to design new similarity metrics which give greater precision to the results offered to users who have cast few votes.

The combination of Jaccard's similarity measure, the arithmetic average of the squared differences in votes and the values of the differences in the votes provide us with the basic elements with which to design a metric that obtains good results in new user cold start situations. These basic elements have been weighted via a linear combination for which the weights are obtained in a process of optimization based on neural learning.

The Jaccard similarity measure makes use of information based on the distribution and on the number of votes cast by each pair of users to be compared. Its use has been a determining factor in achieving the quality of the results obtained, which confirms that it is appropriate to combine this information with traditional information, based on numerical values of the votes, when we wish to design a cold-start similarity measure.

The proposed metric and a complete set of similarity measures have been tested on the Netflix and Movielens databases. The proposed cold-start similarity measure provides results that improve the prediction quality measure MAE and the recommendation quality measures precision & recall. The coverage is the only quality measure that displays inferior result as it is evaluated with the proposed measure; this is due to the fact that the Jaccard component gives priority as neighbors to users with a similar number of votes to the active user.

The proposed similarity measure runs seven times faster than the PIP one, and it also improves the MAE, precision and recall quality results.

In RS, in general, it is feasible to use different metrics on different users, and, in particular, it is possible to use one similarity measure with those users who have cast few votes and a different one on the rest of the users, which enables an improvement in the new users' recommendations without affecting the correct global operation of the RS.

Acknowledgement

Our acknowledgement to the *FilmAffinity.com* & *Netflix companies*, to the Movielens group and to the Elsevier *Knowledge Based Systems journal*.

References

[1] E. 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] N. Antonopoulus, J. Salter, CinemaScreen recommender agent: combining collaborative and content-based filtering, IEEE Intelligent Systems (2006) 35–41.

[3] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1998, pp. 43–52.

[4] J. Bobadilla, F. Serradilla, A. Hernando, Collaborative filtering adapted to recommender systems of e-learning, Knowledge Based Systems 22 (2009) 261–265, doi:10.1016/j.knosys.2009.01.008.

[5] 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, doi:10.1016/j.knosys.2010.03.009.

[6] J. Bobadilla, A. Hernando, F. Ortega, J. Bernal, A framework for collaborative filtering recommender systems. Expert Systems with Applications, in press, doi:10.1016/j.eswa.2011.05.021.

[7] J. Bobadilla, F. Ortega, A. Hernando, A Collaborative Filtering Similarity Measure Based on Singularities, Inf. Proc. and Manag., in press, doi:10.1016/j.ipm.2011.03.007.

[8] J. Bobadilla, F. Ortega, A. Hernando, J. Alcalá, Improving collaborative filtering recommender systems results and performance using genetic algorithms, Knowledge-Based Systems 24 (8) (2011) 1310–1316.

[9] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, Generalization of Recommender Systems: Collaborative Filtering Extended to Group of Users and Restricted to Group of Items, Expert Systems with Applications, in press, doi:10.1016/j.eswa.2011.07.005.

[10] T. Chen, L. He, Collaborative filtering based on demographic attribute vector, in: Proceedings of the International Conference on Future Computer and Communication, 2009, pp. 225–229, doi:10.1109/FCC.2009.68.

[11] C. Christakou, A. Stafylopatis, A hybrid movie recommender system based on neural networks, in: International Conference on Intelligent Systems Design and Applications, 2005, pp. 500–505, doi:10.1109/ISDA.2005.9.

[12] H. Denis, Managing collaborative learning processes, e-learning applications, in: 29th International Conference on Information Technology Interfaces, 2007, pp. 345–350.

[13] L. Ding, D. Steil, B. Dixon, A. Parrish, D. Brown, A relation context oriented approach to identify strong ties in social networks, Knowledge-Based Systems 24 (8) (2011) 1187–1195.

[14] L.Q. Gao, C. Li, Hybrid personalized recommended model based on genetic algorithm, in: International Conference on Wireless Communications, Networking and Mobile Computing, 2008, pp. 9215–9218.

[15] C. Hayes, P. Cunningham, Context boosting collaborative recommendations, Knowledge Based Systems 17 (2–4) (2004) 131–138.

[16] N.M. Heung, E.S. Abdulmotaleb, S.J. Geun, Collaborative error-reflected models for cold-start recommender systems, Decision Support Systems, in press, doi:10.1016/j.dss.2011.02.015.

[17] J.L. Herlocker, J.A. Konstan, J.T. Riedl, L.G. Terveen, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (1) (2004) 5–53.

[18] F. Hernandez, E. Gaudioso, Evaluation of recommender systems: a new approach, Expert Systems with Applications (2007) 790–804, doi:10.1016/j.eswa.2007.07.047.

[19] Z. Huang, D. Zeng, H. Chen, A comparison of collaborative-filtering recommendation algorithms for e-commerce, IEEE Intelligent Systems (2007) 68–78.

[20] Y.P. Huang, W.P. Chuang, Y.H. Ke, F.E. Sandnes, Using back-propagation to learn association rules for service personalization, Expert Systems with Applications 35 (2008) 245–253, doi:10.1016/j.eswa.2007.06.035.

[21] M.H. Hsu, Proposing an ESL recommender teaching and learning system, Expert Systems with Applications 34 (3) (2008) 2102–2110.

[22] J.A. Hyung, A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem, Information Sciences 178 (2008) 37–51, doi:10.1016/j.ins.2007.07.024.

[23] H. Jinghua, W. Kangning, F. Shaohong, A survey of e-commerce recommender Systems, in: International Conference on Service Systems and Service Management, 2007, pp. 1–5, doi:10.1109/ICSSSM.2007.4280214.

[24] H.N. Kim, A.T. Ji, I. Ha, G.S. Jo, Collaborative filtering based on collaborative tagging for enhancing the quality of recommendations, Electronic Commerce Research and Applications 9 (1) (2010) 73–83, doi:10.1016/j.elerap.2009.08.004.

[25] J.A. Konstan, B.N. Miller, J. Riedl, PocketLens: toward a personal recommender system, ACM Transactions on Information Systems 22 (3) (2004) 437–476.

[26] B. Krulwich, Lifestyle finder: intelligent user profiling using large-scale demographic data, Artificial Intelligence Magazine 18 (2) (1997) 37–45.

[27] X.N. Lam, T. Vu, T.D. Le, A.D. Duong, Addressing cold-start problem in recommendation systems, in; Conference On Ubiquitous Information Management And Communication, 2008, pp. 208–211, doi:10.1145/1352793.1352837.

[28] K. Lang, NewsWeeder: learning to filter netnews, in: Proceedings of the 12th International Conference on Machine Learning, 1995, pp. 331–339.

[29] M. Lee, Y. Woo, A hybrid recommender system combining collaborative filtering with neural network, Lecture Notes on Computer Sciences 2347 (2002) 531–534.

[30] C.W. Leung, S.C. Chan, F.L. Chung, An empirical study of a cross-level association rule mining approach to cold-start recommendations, Knowledge Based Systems 21 (7) (2008) 515–529. Autumn.

[31] Q. Li, S.H. Myaeng, B.M. Kim, A probabilistic music recommender considering user opinions and audio features, Information Processing & Management 43 (2) (2007) 473–487.

[32] S.G. Li, L. Shi, L. Wang, The agile improvement of MMORPGs based on the enhanced chaotic neural network, Knowledge Based Systems 24 (5) (2011) 642–651.

[33] S. Loh, F. Lorenzi, R. Granada, D. Lichtnow, L.K. Wives, J.P. Oliveira, Identifying similar users by their scientific publications to reduce cold start in recommender systems, in: Proceedings of the 5th International Conference on Web Information Systems and Technologies (WEBIST2009), 2009, pp. 593–600.

[34] L. Martinez, L.G. Perez, M.J. Barranco, Incomplete preference relations to smooth out the cold-start in collaborative recommender systems, in: Proceedings of the 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009), 2009, pp. 1–6, doi:10.1109/NAFIPS.2009.5156454.

[35] A. Nocera, D. Ursino, An Approach to Providing a User of a "Social Folksonomy" with Recommendations of Similar Users and Potentially Interesting Resources, Knowledge Based Systems 24 (8) (2011) 1277–1296.

[36] M.P. O'Mahony, B. Smyth, A classification-based review recommender, Knowledge Based Systems 23 (4) (2010) 323–329.

[37] S.T. Park, D.M. Pennock, O. Madani, N. Good, D. Coste, Naïve filterbots for robust cold-start recommendations, in: Proceedings of Knowledge Discovery and Data Mining (KDD2006), 2006, pp. 699–705.

[38] Y.J. Park, A. Tuzhilin, The long tail of recommender systems and how to leverage it, in: ACM Conference on Recommender Systems, 2008, pp. 11–18, doi:10.1145/1454008.1454012.
[39] S.T. Park, W. Chu, Pairwise Preference Regression for Cold-start Recommendation, in: ACM Conference on Recommender Systems, 2009, pp. 21–28, doi:10.1145/1639714.1639720.
[40] 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.
[41] P. Pu, L. Chen, Trust-inspiring explanation interfaces for recommender systems, Knowledge Based Systems 20 (6) (2007) 542–556.
[42] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan, J. Riedl, Getting to know you: learning new users preferences in recommender systems, in: International Conference on Intelligent Users Interfaces (IUI2002), 2002, pp. 127–134.
[43] A.M. Rashid, G. Karypis, J. Riedl, Learning preferences of new users in recommender systems: an information theoretic approach, Knowledge Discovery and Data Mining (KDD2008) 10 (2) (2008) 90–100.
[44] L. Ren, L. He, J. Gu, W. Xia, F. Wu, A hybrid recommender approach based on Widrow–Hoff learning, in: International Conference on Future Generation Communication and Networking, 2008, pp. 40–45, doi:10.1109/FGCN.2008.48.
[45] T.H. Roh, K.J. Oh, I. Han, The collaborative filtering recommendation based on SOM cluster-indexing CBR, Expert Systems with Applications 25 (2003) 413–423, doi:10.1016/S0957-4174(03)00067-8.
[46] P.B. Ryan, D. Bridge, Collaborative recommending using formal concept analysis, Knowledge Based Systems 19 (5) (2006) 309–315.
[47] J. Salter, N. Antonopoulus, CinemaScreen recommender agent: combining collaborative and content-based filtering, IEEE Intelligent Systems 21 (2006) 35–41.
[48] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, The Adaptive Web, LNCS 4321 (2007) 291–324.
[49] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, SIGIR (2002) 253–260.
[50] M. Saranya, T. Atsuhiro, Hybrid recommender systems using latent features, in: Proceedings of the International Conference on Advanced Information Networking and Applications Workshops, 2009, pp. 661–666, doi:10.1109/WAINA.2009.122.
[51] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, Providing justifications in recommender systems, IEEE Transactions on Systems, Man and Cybernetics 38 (6) (2008) 1262–1272.
[52] H.F. Wang, Ch.T. Wu, A strategy-oriented operation module for recommender Systems in e-commerce, E-commerce Computers & Operations Research, in press, doi:10.1016/j.cor.2010.03.011.
[53] L.T. Weng, Y. Xu, Y. Li, R. Nayak, Exploiting item taxonomy for solving cold-start problem in recommendation making, in: Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI2008), Dayton, USA, 2008, pp. 113–120.
[54] B. Widrow, M.E. Hoff, Adaptive switching circuits, New York. Convention Record, IRE WESCON, 1960, pp. 96–104.
[55] J.M. Yang, K.F. Li, Recommendation based on rational inferences in collaborative filtering, Knowledge Based Systems 22 (1) (2009) 105–114.
[56] W. Yuan, D. Guan, Y.K. Lee, S. Lee, S.J. Hur, Improved trust-aware recommender system using small-worldness of trust Networks, Knowledge-Based Systems 23 (3) (2010) 232–238.
[57] M.L. Yung, P.K. Chien, TREPPS: a trust-based recommender system for peer production services, Expert Systems with Applications 36 (2) (2009) 3263–3277.