A collaborative filtering approach to mitigate the new user cold start problem
Jesús Bobadilla, Fernando Ortega, Antonio Hernando, Jesús Bernal

ABSTRACT
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) [38] 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-start 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.movie lens.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 first 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 among 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
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. Kohonen'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 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 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
4.1. Formalization

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

\[
\begin{align*}
U &= \{ u \in \text{NaturalNumber} | u \in \{1, L\} \}, \text{ set of users} \quad (1) \\
I &= \{ i \in \text{NaturalNumber} | i \in \{1, M\} \}, \text{ set of items} \quad (2) \\
V &= \{ v \in \text{NaturalNumber} | \text{min} \leq v \leq \text{max} \} \cup \{ * \}, \text{ set of possible votes} \quad (3) \\
R_u &= \{(i, v) | i \in I, v \in V \}, \text{ ratings of user } u \quad (4) \\
\text{We define vote } v \text{ of user } u \text{ on item } i \text{ as } r_{uv} = v \\
\text{We define the average of the valid votes of user } u \text{ as } r_u
\end{align*}
\]

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 \([\text{min}, \text{max}]\), where the absence of ratings will be represented by the symbol \(*\).
We define the cardinality of a set C as its number of valid elements

\[ |C| = \# \{ x \in C | x \neq \bullet \} \]

In this way:

\[ |R_u| = \# \{ i \in R_u | i \neq \bullet \} \]  

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

4.2. Set of basic similarity measures

In order to find the similarity 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. \( v^0 \), \( v^1 \), \( v^2 \), \( v^3 \) (Eqs. (9)–(11)).
  2. Mean squared differences \( \sigma \), (Eqs. (12) and (13)).
  3. Standard deviation of the squared differences \( \sigma \), (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.

\( v^1 \) 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.

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

\( \mu \) provides the simplest and most intuitive measure of similarity between users, but it could be a good idea to complement it with the importance held by the extreme cases in this measure, using \( \sigma \).

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.

\[ V_{xy}^d = \{ i \in I | r_{ix} \neq \bullet \land r_{iy} \neq \bullet \land |r_{ix} - r_{iy}| = d \} \text{ where } d \in \{0, \ldots, \max - \min\} \]  

We define \( d \in \{0, \ldots, \max - \min\} \), \( b_{xy}^d = |V_{xy}^d| \) (10)

finally, we perform the normalization:

\[ t_{xy}^d = \frac{b_{xy}^d}{\sum_{d=0}^{\max - \min} b_{xy}^d}, \quad s_{xy}^d \in [0,1] \]  

Let \( C_{xy} = \{ i \in I | r_{ix} \neq \bullet \land r_{iy} \neq \bullet \} \), the set of items voted simultaneously by both users

\[ \mu_{xy} = 1 - \frac{1}{|C_{xy}|} \sum_{i \in C_{xy}} \left( \frac{|r_{ix} - r_{iy}|}{\max - \min} \right)^2 \implies \mu_{xy} \in [0,1] \]  

\[ \sigma_{xy} = \sqrt{\frac{1}{|C_{xy}|} \sum_{i \in C_{xy}} \left( \frac{|r_{ix} - r_{iy}|}{\max - \min} \right)^2 - \left( \frac{1}{|C_{xy}|} \sum_{i \in C_{xy}} \left( \frac{|r_{ix} - r_{iy}|}{\max - \min} \right)^2 \right)} \implies \sigma_{xy} \in [0,1] \]  

Let \( R_{xy} = \{ i \in I | r_{ix} \neq \bullet \} \), the set of items voted by user \( u \)

\[ jaccard_{xy} = \frac{|R_{xy} \cap R_{xy}^c|}{|R_{xy} \cup R_{xy}^c|} \implies jaccard_{xy} \in [0,1] \]  

4.3. Similarity measures selected

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

\[ \{ v^0, v^1, v^2, v^3, \sigma, \mu, jaccard \} \]  

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

\[ \text{Table 2 Measures.} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Measures descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{xy}^d )</td>
<td># Items with the same value in user x and user y (normalized)</td>
</tr>
<tr>
<td>( t_{xy}^1 )</td>
<td># Items with a difference of 1 stars in user x and user y (normalized)</td>
</tr>
<tr>
<td>( t_{xy}^2 )</td>
<td># Items with a difference of 2 stars in user x and user y (normalized)</td>
</tr>
<tr>
<td>( t_{xy}^3 )</td>
<td># Items with a difference of 4 stars in user x and user y (normalized)</td>
</tr>
<tr>
<td>( \mu_{xy} )</td>
<td>Mean squared differences (user x, user y)</td>
</tr>
<tr>
<td>( \sigma_{xy} )</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>Jaccard_{xy}</td>
<td>Jaccard similarity measure</td>
</tr>
<tr>
<td>( w_{xy} )</td>
<td>Similarity measure weights</td>
</tr>
<tr>
<td>( p_{xy} )</td>
<td>Prediction to the user on the item</td>
</tr>
<tr>
<td>( m_{xy} )</td>
<td>Prediction error on user x, item i</td>
</tr>
<tr>
<td>( m_u )</td>
<td>User u mean absolute error</td>
</tr>
<tr>
<td>( m_c )</td>
<td>RS mean absolute error</td>
</tr>
<tr>
<td>( e_{xy} )</td>
<td>User u, item i, coverage</td>
</tr>
<tr>
<td>( c_u )</td>
<td>User u coverage</td>
</tr>
<tr>
<td>( q_{xy} )</td>
<td>RS coverage</td>
</tr>
<tr>
<td>( i_{xy} )</td>
<td>Is i recommended to the user u?</td>
</tr>
<tr>
<td>( i_{xy}^c )</td>
<td>Is i recommended relevant to the user u?</td>
</tr>
<tr>
<td>( t_{xy} )</td>
<td>Prediction of the user u</td>
</tr>
<tr>
<td>( t )</td>
<td>Prediction of the RS</td>
</tr>
<tr>
<td>( n_{xy} )</td>
<td>Is i not recommended relevant to the user u?</td>
</tr>
<tr>
<td>( n_u )</td>
<td>Recall of the user u</td>
</tr>
<tr>
<td>( x )</td>
<td>Recall of the RS</td>
</tr>
</tbody>
</table>

\[ \text{Table 3 Sets.} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Sets descriptions</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U )</td>
<td>Users</td>
<td>L</td>
</tr>
<tr>
<td>( I )</td>
<td>Items</td>
<td>M</td>
</tr>
<tr>
<td>( V )</td>
<td>Rating values</td>
<td>user, user</td>
</tr>
<tr>
<td>( R_u )</td>
<td>User ratings</td>
<td>user</td>
</tr>
<tr>
<td>( v_{xy}^d )</td>
<td>Items rated with a difference of d stars</td>
<td>user x, user y</td>
</tr>
<tr>
<td>( G_{xy} )</td>
<td>Items rated simultaneously by users x and y</td>
<td>user x, user y</td>
</tr>
<tr>
<td>( K_u )</td>
<td>Items voted by user u</td>
<td>user</td>
</tr>
<tr>
<td>( K )</td>
<td>Users neighborhoods of the user</td>
<td>user, k</td>
</tr>
<tr>
<td>( H_{xy} )</td>
<td>Users neighborhoods which have rated the item</td>
<td>user, i, k</td>
</tr>
<tr>
<td>( X_{xy} )</td>
<td>Top recommended items to the user</td>
<td>user, i, k</td>
</tr>
<tr>
<td>( Z_{xy} )</td>
<td>Top N recommended items to the user</td>
<td>user, k, N, \theta</td>
</tr>
<tr>
<td>( U^v )</td>
<td>Training users</td>
<td>user</td>
</tr>
<tr>
<td>( U^v )</td>
<td>Validation users</td>
<td>user</td>
</tr>
<tr>
<td>( f^v )</td>
<td>Training items</td>
<td>user</td>
</tr>
<tr>
<td>( f^v )</td>
<td>Validation items</td>
<td>user</td>
</tr>
<tr>
<td>( M_u )</td>
<td>Items rated by user u where a prediction can be determined</td>
<td>user</td>
</tr>
<tr>
<td>( O_u )</td>
<td>Validation users with assigned MAE</td>
<td>user</td>
</tr>
<tr>
<td>( C_u )</td>
<td>Validation users with assigned coverage value</td>
<td>user</td>
</tr>
<tr>
<td>( O^v )</td>
<td>Validation users with assigned coverage value</td>
<td>user</td>
</tr>
<tr>
<td>( S )</td>
<td>Validation users with assigned precision value</td>
<td>user</td>
</tr>
<tr>
<td>( V_u )</td>
<td>Set of recommended relevant items (true-positives)</td>
<td>user</td>
</tr>
<tr>
<td>( N_u )</td>
<td>Set of not recommended relevant items</td>
<td>user</td>
</tr>
<tr>
<td>( S )</td>
<td>Validation users with assigned recall value</td>
<td>user</td>
</tr>
</tbody>
</table>
4.5. Neural network learning

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):

\[ \{v^0, v^1, v^2, v^4, \mu, \text{Jaccard}\} \]  

(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:

\[ \text{MJD}_{xy} = \frac{1}{6} \left( w_1 v^0_{xy} + w_2 v^1_{xy} + w_3 v^2_{xy} + w_4 v^4_{xy} + w_5 \mu_{xy} + w_6 \text{Jaccard}_{xy} \right) \]  

(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 experiments; through this methodology, we calculate the quality results of the predictions and recommendations for the similarity measures studied.

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

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 different k-neighbors.

5.1. Obtaining prediction and recommendations

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

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Weights obtained using the gradient descent method (Netflix RS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1 )</td>
<td>0.66</td>
</tr>
<tr>
<td>( w_2 )</td>
<td>0.35</td>
</tr>
<tr>
<td>( w_3 )</td>
<td>-0.21</td>
</tr>
<tr>
<td>( w_4 )</td>
<td>-0.43</td>
</tr>
<tr>
<td>( w_5 )</td>
<td>1.07</td>
</tr>
<tr>
<td>( w_6 )</td>
<td>0.31</td>
</tr>
</tbody>
</table>

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

\[ \text{MAE}^{\text{CF}}(\text{MJD}(\text{Netflix})) \] 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 \( \text{MAE}_{\text{MJD}} \) is an upper bound of the accuracy obtained in each instant \( t \) \( \text{MAE}^{\text{CF}}(\text{MJD}(\text{Netflix})) \) 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. The error measure, we use the system \( \text{MAE}_{\text{MJD}} \), making use of a very small set of test users with the aim of achieving reasonable execution times. These calculations are obtained by parallel processing through a cluster of computers.

The learning of the neuronal network is carried out using a set of pairs of users \( (u_i, u_j) \) where \( u_i \) represents a cold-start user (who has rated between 2 and 20 items) and \( u_j \) represents any user in the database. This pair of users are taken into account for updating the weights \( w_i \).
The equation proposed in (27). The predictions obtained are sum-

\[ K_u \subseteq U \land \# K_u = k \land u \notin K_u \]

(21)

\[ \forall x \in K_u, \forall y \in (U - K_u), \text{sim}_{u,x} > \text{sim}_{u,y} \]

(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:

Let \( P_u = \{(i,p) | i \in I, p \in \text{RealNumber}\} \)

set of prediction to the user u

We will assign the value of the prediction \( p \)

made to user u on item i as \( p_{ui} = p \)

(24)

Once the set of k users (neighbors) similar to active user 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)).

\[ H_{aj} = \{ n \in K_u | r_{u,n} \neq \bullet \} \]

(25)

\[ p_{ui} = r_{u,i} + \frac{1}{\sum_{n \in H_{aj}} \text{sim}_{u,n}} \sum_{n \in H_{aj}} \text{sim}_{u,n} (r_{u,n} - \bar{r}) \Leftrightarrow H_{aj} \neq \emptyset \]

(26)

\[ p_{ui} = \bullet \Leftrightarrow H_{aj} = \emptyset \]

(27)

5.1.3. Top N recommendations

We define \( R_u \) as the set of top N recommendations to user u.

The following must hold:

\[ R_u \subseteq I \forall i \in R_u, \quad r_{ui} = \bullet, \quad p_{ui} \neq \bullet, \quad \text{sim}_{u,i} = \text{sim}_{u,j} \]

(28)

\[ Z_u \subseteq X_u, \# Z_u = N, \quad \forall x \in Z_u, \quad \forall y \in X_u : p_{u,x} > p_{u,y} \]

(29)

If we want to impose a minimum recommendation value: \( \theta \) \in \text{Real-Number}, we add \( p_{ui} \geq \theta \)

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:

\[ U = \{ u_1, u_2, u_3, u_4, u_5, u_6 \} \]

\[ I = \{ i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11}, i_{12} \} \]

\[ V = \{ 1, 2, 3, 4, 5, \bullet \} \]

Table 5 shows the votes \((r_{ui})\) 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

<table>
<thead>
<tr>
<th>( r_{ui} )</th>
<th>( i_1 )</th>
<th>( i_2 )</th>
<th>( i_3 )</th>
<th>( i_4 )</th>
<th>( i_5 )</th>
<th>( i_6 )</th>
<th>( i_7 )</th>
<th>( i_8 )</th>
<th>( i_9 )</th>
<th>( i_{10} )</th>
<th>( i_{11} )</th>
<th>( i_{12} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>( u_3 )</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>( u_4 )</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>( u_5 )</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>( u_6 )</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

If we want to make \( N = 3 \) recommendations: \( Z_u = \{ i_7, i_9, i_8 \} \).

### Table 6

<table>
<thead>
<tr>
<th>MJD</th>
<th>( u_2 )</th>
<th>( u_3 )</th>
<th>( u_4 )</th>
<th>( u_5 )</th>
<th>( u_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>1.166</td>
<td>1.155</td>
<td>1.415</td>
<td>1.572</td>
<td>0.887</td>
</tr>
</tbody>
</table>

### Table 7

Predictions that \( u_2 \) can receive using MJD, \( k = 2 \) and arithmetic average as aggregation approach.

<table>
<thead>
<tr>
<th>( r_{ui} )</th>
<th>( i_1 )</th>
<th>( i_2 )</th>
<th>( i_3 )</th>
<th>( i_4 )</th>
<th>( i_5 )</th>
<th>( i_6 )</th>
<th>( i_7 )</th>
<th>( i_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4.5</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2.5</td>
</tr>
</tbody>
</table>

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 appropriate 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.

\[ U' \subset U, \text{ set of training users} \]

\[ U'' \subset U, \text{ set of validation users} \]

\[ I' \subset I, \text{ set of training items} \]

\[ I'' \subset I, \text{ set containing the validation item} \]

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

\[ U'' \cup U' = U, \quad U'' \cap U' = \emptyset, \quad \# I' = 1, \quad I' \cap I'' = I, \quad I'' \cap I' = \emptyset \]

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)).

\[
C_{ui} = \{i \in [2, \ldots, 20] \land r_{ui} \neq \bullet \}
\]

We define the system’s MAE as:

\[
\text{The MAE of the RS: } m = \frac{1}{|O|} \sum m_u, \quad m_u \in [0, \max - \min]
\]

The MAE of the user \( u \):

\[
\text{The coverage of the RS: } c = \frac{1}{|U|} \sum C_u, \quad C_u \in [0, \max - \min]
\]

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:

Let \( S = \{u \in U^r | t_u \neq \bullet \} \)

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

\[
c_{ui} = \begin{cases} 1 \quad u \in U^r, \ i \in I^r, \ p_{ui} \neq \bullet \land r_{ui} = \bullet \\ 0 \quad \text{otherwise} \end{cases}
\]

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

\[
C_u = \{i \in I^r | c_{ui} \neq \bullet \quad \text{where } u \in U^r\}
\]

The coverage of the user \( c_u \) 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.

\[
c_u = 100 \times \frac{|C_u|}{|U|} \quad \text{where } u \in [0, 100]
\]

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

\[
c = \frac{1}{|U|} \sum c_u, \quad c \in [0, 100]
\]

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 \( \theta \) as the minimum value of a vote to be considered relevant.

Let \( u \in U^r \land i \in I^r \land r_{ui} \neq \bullet \)

\[
l^r = \{i \}, \ l = \{j \in I^r | p_{ui} \neq \bullet \land j \neq i\}
\]

Through \( l^r \) we obtain \( k \)-neighbors (\( K_u \)), and through \( K_u \) and \( l^r \) we make the prediction \( p_{ui} \).

Each \( q_{ui} \) term indicates whether item \( i \) has been recommended to user \( u \).

\[
q_{ui} = \begin{cases} 1 \quad u \in U^r, i \in l^r \land r_{ui} \neq \bullet \land \theta > \text{recommended item} \\ 0 \quad \text{otherwise} \end{cases}
\]

Each \( t_{ui} \) term indicates whether item \( i \) recommended to user \( u \) has been relevant.

\[
t_{ui} = \begin{cases} 1 \quad u \in U^r, i \in l^r \land q_{ui} \neq \bullet \land \theta > \text{relevant} \\ 0 \quad \text{otherwise} \end{cases}
\]

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

\[
\alpha_u = \frac{\# \{i \in I_t | q_{ui} \neq \bullet \} \land \theta > \text{relevant} \} \neq \emptyset, \quad \alpha_u \in [0, 1]
\]

\[
\alpha = \frac{\# \{i \in I_t | q_{ui} \neq \bullet \} \neq \emptyset, \quad \alpha \in [0, 1]
\]

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

\[
\text{The precisión of the RS: } t = \frac{1}{|U|} \sum t_u, \quad t_u \in [0, 1]
\]
We define the system's precision as:

\[ t = \frac{1}{\#S} \sum_{i \in S} f_i \iff S \neq \emptyset, \quad t \in [0, 1] \quad (63) \]

\[ t = \emptyset \iff S = \emptyset \quad (64) \]

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 \( \theta \) as the minimum value of a vote to be considered as relevant.

Let \( u \in U \), \( i \in I \), \( u \in I \) \( \#R_u \in [2, \ldots, 20] \), and \( R_u \neq \emptyset \).

\[ I' = \{ i \}, \quad P' = \{ j \in I \mid R_u \neq \emptyset \} \quad (65) \]

With \( f' \) we obtain k-nearest neighbors (\( K_u \)) and with \( K_u \) and \( f' \) we make prediction \( p_u \).

Each term \( n_u \) indicates whether item \( i \) not recommended to the user \( u \) is relevant.

\[ n_u = \emptyset \iff u \in U', \quad i \in I', \quad p_u = \emptyset \iff i \notin R_u, \quad \text{not rec.} \quad \text{or} \quad n_u = \emptyset \iff u \in U', \quad i \in I', \quad p_u = \emptyset \iff i \notin R_u, \quad \text{not rec.} \quad \text{or} \quad n_u = \emptyset \quad (66) \]

We define \( Y_u \) as set of relevant items to user \( u \) which have been relevant.

\[ Y_u = \{ i \in I \mid n_u \neq \emptyset \}, \quad u \in U' \quad (67) \]

We define \( N_u \) as set of not recommended items to user \( u \) which have been relevant.

\[ N_u = \{ i \in I \mid n_u \neq \emptyset \}, \quad u \in U' \quad (68) \]

The recall of the user \( u(x) \) 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).

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

\[ x_u = \emptyset \iff Y_u \cup N_u = \emptyset \quad (70) \]

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

\[ S = \{ u \in U' \mid n_u \neq \emptyset \} \quad (71) \]

We define the system's recall as:

\[ x = \frac{1}{\#S} \sum_{u \in S} x_u \iff S \neq \emptyset, \quad x \in [0, 1] \quad (72) \]

\[ x = \emptyset \iff S = \emptyset \quad (73) \]

5.2.5. Running example

We establish \( U' = \{ u_1, u_2 \} \), \( U' = \{ u_3, u_4, u_5, u_6 \} \), \( k = 2 \).

Table 8 shows an outline of the process with which we obtain the quality measures MAE, precision and recall of user \( u_1 \). The two first columns \( (I' \) and \( f' \) \) describe respectively the validation and training items. The third column \( (\text{MJD}_{u_1, u_2}) \) informs about the similarity between \( u_1 \) and each validation user. The fourth column \( (K_u) \) describes the k-neighbors of \( u_1 \). The fifth column \( (P_{u_1, u_2}) \) shows the prediction of item \( i \) on user \( u_1 \) using the arithmetic mean as aggregation approach instead of Eq. (27). The sixth column \( (r_{u_1, u_2}) \) informs about the vote of \( U_1 \) 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 \((q^+\neq)\) and when it has been recommended and it is relevant \((t^+\neq)\); the column "Recall" shows when a relevant item has been recommended \((t^+\neq)\), and when it has not \((n^+\neq)\).

In the example, we make the following calculations which determine the similarity between users \( u_1 \) and \( u_3 : \text{MJD}_{u_1, u_3} = 1.166, \text{MJD}_{u_1, u_4} = 1.415, \text{MJD}_{u_1, u_5} = 1.572, \text{MJD}_{u_1, u_6} = 0.887, \text{MJD}_{u_2, u_3} = 0.5 \) with \( \text{Jaccard}_{u_1, u_3} = 2/7 = 0.286, \text{Jaccard}_{u_1, u_4} = 2/7 = 0.286, \text{Jaccard}_{u_1, u_5} = 2/7 = 0.286, \text{Jaccard}_{u_1, u_6} = 2/7 = 0.286 \).

Table 9 specifies the way to obtain the coverage of \( u_1 \) with each of its 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
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: 480,189 users, 17,770 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 \( \#R_v \in [2, \ldots, 20] \). We use the fixed value of k-neighbors \( k = 700 \) and number of recommendations \( N = 10 \).
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
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 Movielens. 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


