Information Sciences 287 (2014) 90–108

Contents lists available at ScienceDirect

Information Sciences

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

Learning from explanations in recommender systems

Sergio Cleger [a,1], J.M. Fernández-Luna [b], Juan F. Huete [b,*]

[a] *Departamento de Informática, Facultad de Informática y Matemática, Universidad de Holguín, Cuba*
[b] *Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática y de Telecomunicación, CITIC-UGR Universidad de Granada, 18071 Granada, Spain*

A R T I C L E I N F O

Article history:
Received 27 June 2013
Received in revised form 11 June 2014
Accepted 24 July 2014
Available online 1 August 2014

Keywords:
Recommender system
Recommendation explanation
Learning

A B S T R A C T

Although recommender systems are extremely useful when it comes to recommending new products to users, it is important that these applications explain their recommendations so that users can consider and trust them. There is also another reason: by analyzing why the system recommends a particular item or proposes a certain rating, the user might also consider the quality of the recommendation and, if appropriate, change its predicted value. On this basis, this paper presents a new technique to improve recommendations based on a series of explanations that should be given when various already-known items are recommended. To summarize, the aim of our proposal is to learn a regression model from the information presented in these explanations and, where appropriate, use this model to change the recommendation for a target item. In order to test this approach, we experimented with the MovieLens data set. A number of lessons can be learned: firstly, it is possible to learn from a set of explanations, although this is highly user-dependent; and secondly, we can use an automatic procedure to analyze the role of the different features presented in an explanation. We consider these results to be interesting and to validate our novel approach.

1. Introduction

Ever since they were first conceived in the last decade of the 20th century [10,32,33], recommender systems (RS) have been used in many different areas. There are currently a large number of RSs on Internet sites such as Amazon, Netflix or Last.fm. These systems are designed to learn from a user's behavior to discover their preferences and either help them find what they are specifically looking for or what they might find useful in a vast amount of information.

Despite the popularity of such systems, there are various reasons why users might feel uneasy about using them and/or relying on their recommendations. These systems are usually seen as black boxes in which there is no other choice than to trust their recommendations [14]. Much research has been conducted to focus on this problem and among the proposed solutions we focus our attention on the use of explanation facilities which have also been considered as a critical function for recommender systems [8,9,20,36]. These facilities provide users with an explanation of the rationale behind such recommendations, enabling them to better understand the workings of the system. Tintarev and Masthoff [41] list a number of reasons why recommender systems should provide such an explanation and these include:

* Corresponding author.
E-mail addresses: sergio@facinf.uho.edu.cu (S. Cleger), jmfluna@decsai.ugr.es (J.M. Fernández-Luna), jhg@decsai.ugr.es (J.F. Huete).
[1] Visiting Academic at the Universidade do Estado do Amazonas, Brazil.

http://dx.doi.org/10.1016/j.ins.2014.07.031
0020-0255/© 2014 Elsevier Inc. All rights reserved.

1. Transparency (illustrates how a recommendation is computed).
2. Trust (increases user confidence).
3. Effectiveness (helps the user make a good decision).
4. Scrutability (allows users to tell the system it is wrong).

An algorithm's ability to explain both recommendation results and the explanation itself strongly depends on the recommender model used, more especially so in the case of memory-based approaches [36] which are based on the similarities extracted from known data and which can serve as the basis for explaining recommendation results. In this paper, we shall assume a nearest-neighborhood-based RS in order to predict the rating for a target item. In this recommendation paradigm, there are two good alternatives for explaining recommendations [15]: (i) the use of a bar chart histogram which explains that "the system suggests 3 stars because it has been rated by other similar users as ..."; and (ii), based on the system's previous performance, i.e."The system correctly recommended for you 80% of the time". This second alternative is valuable for increasing user trust but does not fulfill any of the other criteria.

Focusing on the first more informative alternative, this partly explains *how* the predicted rating is computed (transparency), for instance by combining the ratings into a weighted average prediction. In this case, we can obtain several explanations for the same recommended rating, let us say 3. If we assume that there are five neighbors (who each contribute equally to the prediction), this rating can either be obtained when the suggested ratings are $\{1,2,3,4,5\}$ or when the ratings are $\{3,3,3,3,3\}$. Users might find both explanations useful but while the user is unsure of the quality of the prediction in the first, the explanation reinforces the recommendation in the second (this is related to trust). In the same way, we can consider a situation where the system predicts the value 4 because the neighbors rated the target item as $\{1,5,5,5,5\}$. The information given in the explanation can then be used to change the prediction by modifying the rating to 5 (fact related to scrutability).

In our previous research [4], we found that having observed an explanation, users often modified their proposed rating (around 35% of the time). Moreover, better predictions about the quality of the proposed ratings were obtained once a large number of explanations had been observed. Certain learned processes must exist in the user's mind to cause them to act on the prediction. In this paper, we talk about such an action in terms of trusting, mistrusting or even possibly changing the predicted rating. This situation underlies our research proposal, which can be formulated as the following question: *Would it be possible to learn from a set of explanations?* If the answer is "yes", we must explore *how the learned information can be used to improve recommendations*. With the aim of learning from explanations, we propose the use of a machine-learning [13] algorithm that is capable of inducing general rules about the user's actions from a set of observed instances.

In order to obtain the set of training instances, a first approach might be to gather the required information from user experience of the system, by either explicitly or implicitly analyzing their actions over the observed explanations. This task involves various problems: (a) user burden: we need to know what action the user takes after observing the explanation for each recommended item even when they have no interest in it; (b) dependence on the recommendation strategy: when considering a neighborhood-based RS, it could be that the set of neighbors used to obtain the predictions (and their associated explanation) varies with the recommended items; and (c) the cold-start problem: we need enough feedback to start the learning process.

In order to solve this problem, we propose that it be tackled from a different perspective: let us consider a given (unobserved) target item, I_t, where N is the set of neighbors selected to compute the pair prediction/explanation. We will use *this* neighborhood to obtain a prediction for all the items rated by the user, let us say $\mathcal{R} = \{I_1, \ldots, I_k\}$, and their associated explanations. For these items, since we know the real and predicted ratings, we can automatically presume what the user's actions should be, such as, for instance, to trust or increase the predicted value. Using this approach, we shall obtain a corpus of tuples $\langle$predictions, explanations and actions$\rangle$ that can be used as training data to induce the particular action for the target item. We shall say that this approach is based on previous explanations.

We would like to stress that the main focus of our paper is to determine whether the general idea of learning from explanations may work. As a result, we conclude that explanations represent a valuable source of information that might be exploited since this is (as far as we know) the first attempt to explore their impact on a recommending environment. By means of this approach, we should explore the costs and benefits of scrutability for improving recommendations, something which is considered to be an open problem in the field [21,23,42]. Although our approach focuses on a neighborhood-based RS, it could also be applied to other recommending strategies, such as those using matrix factor [22] or additional informational elements such as the quality of the items [40] or social-based knowledge [35].

This paper is organized as follows: the next section discusses the reasons for our approach; Section 3 presents related work on recommender systems; Section 4 describes our approach; Section 5.1 outlines empirical experimentation with MovieLens data sets; and finally Section 6 presents our concluding remarks.

2. Explaining by using predictions for already known items

As we mentioned previously, our research stems from the use of predictions on a set of observed items which serve as the basis for explanation facilities [4]. More specifically, this research considered a collaborative RS to predict how an active user, a, might rate a target item, t, $\hat{r}(a,t)$. Following a user-based approach, the predictions are obtained using a weighted combination of the suggestion given by the nearest N neighbors, i.e.

$$\hat{r}(a,t) = \sum_{v \in N} w(v) sg(v,t), \tag{1}$$

where $w(v)$ represents the importance of each neighbor v and $sg(v,t)$ the suggestion of this neighbor to the target item, which can be defined as a normalizing function of the rating given by each neighbor, $r(v,t)$. For example, $sg(v,t) = \bar{r}(a) + (r(v,t) - \bar{r}(v))$, with $\bar{r}(\cdot)$ representing the user's mean rating.

In this context, if we ask the system "Why are you recommending me this rating?", the system might say ...

because the average rating given to this item by other users with similar tastes to yours was 3★.

It is not, however, possible for users to infer from this kind of explanation anything about the reliability of a recommendation.

In order to begin our explanation approach, we shall first consider a simple example:

Let us suppose that a group of people recommend you see a particular film, e.g. *A*, (or go to a certain restaurant, hotel, etc.) because their average rating is "good". In order for you to reach a decision about the movie, you ask them to justify their given recommendation. By way of explanation, it might be natural for each person to support their opinion using movies they have previously seen. Someone might say, therefore, that they recommended it because it is similar to *B* which they enjoyed very much. However, you might also consider their opinion about dissimilar items (and whether they liked them or not) to be relevant for your decision making. This is because if you are interested in detecting a wrong recommendation, it would be helpful to analyze conflicting opinions. After amassing all the arguments from all your friends, you will have enough knowledge to make your own opinion about the quality of the recommendation.

Our hypothesis is that there are two main reasons why this type of opinion is valuable for users: the first is that the opinion comes from like-minded users ("friends"); and the second is that we contextualize the explanations using the active user's own background (items already known).

The implementation of this idea into our framework will therefore mean that justifying a recommendation will also involve taking into account how the neighbors rated the items which were previously rated by the active user, i.e. neighborhood opinion for these items. For each item, we can obtain two different pieces of information. The first concerns whether the neighbors will be able to properly predict how the user will rate this item. In this case, by taking into account the quality of the different predictions we could obtain a measure of neighborhood confidence. The second concerns us knowing the similarity between a rated item and the target one in terms of the ratings given by the neighbors to both the observed and target items. By combining both pieces of information we could obtain some idea of the trustworthiness of the neighborhood with respect to the target item.

We have therefore designed and evaluated a general explanation interface [4] which includes various pieces of information that we consider to be useful for explaining how the neighbors rated the target item in particular but also information relating the neighborhood to the user's background. More formally, the main relevant components comprising the graphical interface for the explanation (see Fig. 1 for an example) in terms of learning from the explanation are:

- *Bar chart histogram*, which show agreement on the prediction: we use a bar chart for each possible category (ratings from 1★ to 5★) to represent the number of neighbors who rated items accordingly (see left-hand side of Fig. 1). In this case, the explanation says that "the system suggests *x*★ because it has been rated as ... by ...". This interface is the most convincing explanation of why a movie was recommended in MovieLens [15].
- *2D past-recommendations*: in order to show the information relating `this` neighborhood with the user's background, we use a 2D graph. More specifically, recommendations for certain previous items are arranged as points on a two-dimensional graph where the *X* axis shows the similarity between each item and the target one and the *Y* axis shows the neighborhood behavior for each previous experience (the quality of the suggestions, i.e. whether neighbors' suggestions match the rating given by the active user to the target item).

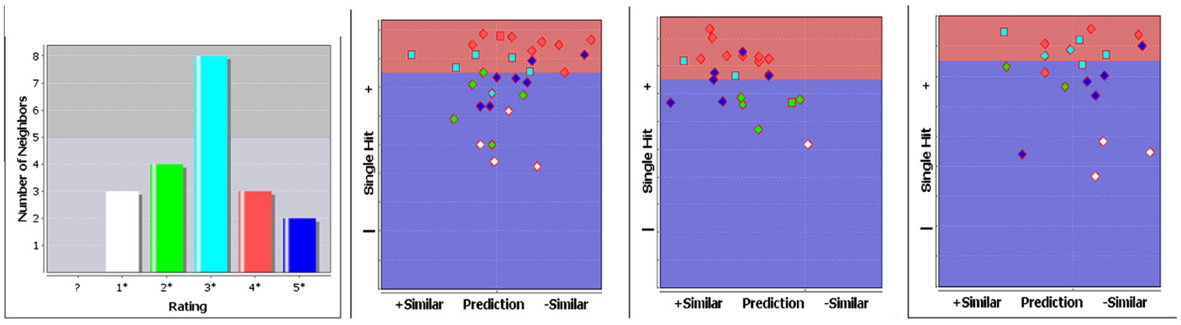

Fig. 1. Histogram and various 2D explanations. The first 2D explanation corresponds to the same target item as the histogram. The last two explanations illustrate how they vary when different target items are considered.

We will now describe in greater detail the pieces of information which are included in the 2D graph (for illustrative purposes, we also present another 2D graph of previous recommendations in Fig. 1):

1. Explanation support, $\mathcal{S}$: this is related to the number of items belonging to the active user's profile that can be recommended by this particular neighborhood. When an item is only rated by a few neighborhood members, we do not consider the predictions to be good enough and so do not include them in the explanation. More specifically, we only consider the predictions for those items rated by at least 70% of `these` neighbors. Explanation support can therefore reflect the similarity between the profiles of different individuals in absolute terms since it may be considered as their (soft) intersection.
2. Position on the grid relating each item $i \in \mathcal{S}$ to the target one, t.

   X: this represents the similarity between the predictions for items $i \in \mathcal{S}$ and t so that those items closer to the target item (items with similar suggestions) will be placed on the left. The location on the X-axis is therefore computed using a distance measure which takes into account the suggestions given by each neighbor, v, to the target item, t, and their suggestions for the known item i, i.e.

   $$d(i,t) \propto \sum_{v \in N} |sg(v,i) - sg(v,t)|. \tag{2}$$

   Fig. 2 shows an explanation for the film "Desperado". The active user can see that, of all the previously rated films, the film with the most similar rating pattern to the one being recommended is "Interview with the Vampire", while the "The Empire Strike Backs" is the most different.

   Y: represents the quality of individual suggestions, i.e. whether each neighbor hits the active user's rating for the item, $r(a,i)$. Any item which has been properly predicted by every neighborhood member will be placed towards the top of the Y-axis. For these items we are aware of a consensus among the individuals' predictions. In particular, we compute the hit value for each item as

   $$h(i) \propto \sum_{v \in N} |r(a,i) - sg(v,i)|. \tag{3}$$

   In Fig. 2, we can see a film located at the lowest value of the Y-axis, which means that there is a large number of neighbors who did not match the active user's rating for this film.
3. Quality of previous recommendations: we can show whether each particular recommendation matches the rating given by the user (we denote this situation with a square $\square$) or not (in which case we use a diamond $\diamond$). Additionally, the color used to fill in these figures represents the rating given to each item by the active user (we use the same colors as those used in the histogram). By hovering over these figures, an informational balloon appears showing the movie in question, the user's real rating and the system's prediction.

The user study which we performed to test the explanation interfaces concluded that those users who can control the recommendations are often more satisfied (even when this control implies greater effort) and this coincides with the opinion given by Konstan and Rield [21]. We also concluded that the histogram and 2D graph present the user with two complementary pieces of information (although users might mistrust an anonymous recommendation obtained through low neighborhood confidence and with no consensus among individuals, they might trust it if there were a set of high quality neighbors).

We also found that by exploring the information in the 2D graph some users were able to correct any error in the recommendations, although there is great variance in the results obtained by different users. Correspondingly, we are aware that when users analyze the given information, they tend to focus on certain parts of the explanation. For example, we found a user who considers explanation support to be highly relevant, particularly for those items on the left-hand side of the 2D graph. In this case, users feel confident when they discover that they have rated many items which are similar to the target

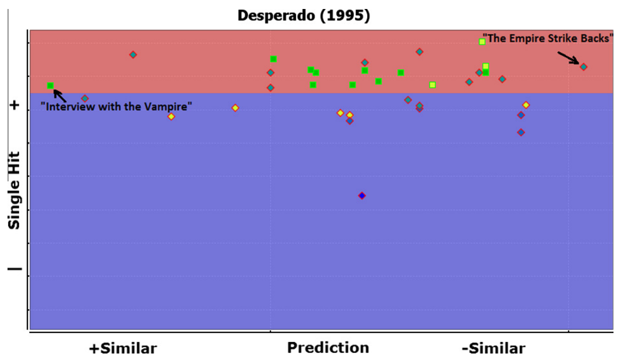

Fig. 2. Example of explanation for Desperado.

one. Moreover, by looking at the performance of these predictions, the user could grasp the quality of target prediction. In particular, we found that the system is prone to overestimating the rating when the support in this area is greater than 10. After analyzing the data (see Fig. 3), we were able to confirm our hypothesis; the support on the left-hand side highly correlates with the error in the recommendations (correlation = 0.58). It might therefore be used to determine the action that should be performed on prediction. For instance, if the system predicts the rating 3.75 for the target item and there are 25 items on the left-hand side of the 2D graph supporting this recommendation, $x = 25$, then the estimated error is 0.71, $error = 0.061x - 0.811$, and the user can modify the recommendation to 3.

We consider these results to be interesting since they represent an additional piece of information that can be included in a recommending model. In this paper, we shall explore the cost and benefits of this inclusion.

3. Related work

The first study to explore explanation in recommender systems as an important research problem was conducted by Herlocker et al. [15]. In this paper, the authors conclude that explanation can persuade users to have greater confidence in the system. The authors conducted an experimental study to address various hypotheses relating to the explanation, proposing 21 different interfaces, playing with certain elements such as a neighbor's rating, overall previous performance, rating similarity or explanation focused on certain items. Since then, various lines of research have been conducted into explanation in RS [1,2,4,6,7,20,30,38,42] with quite different aims. While all highlight the need to explain recommendations, explanations in recommender systems are still considered an open research problem [21], a field in which additional work is needed in order to evaluate their multiple impact.

In their excellent review on explanation, Tintarev and Masthoff [41], mention seven criteria that are related to the aims generally pursued with RSs: explaining how the algorithm works (transparency); providing users with the ability to certify the recommendation (scrutability), or helping them making quick (efficiency), good or appropriate (effectiveness) decisions; pursuing the goal of increasing the user's enjoyment (satisfaction) or confidence in the system (trust), or convincing them to acquire, purchase or select an option from the available collection (persuasiveness).

Most approaches use a user study to measure the impact of explanations from the user's point of view, for instance by helping the user make good (accurate) decisions [1]. In such situations, the effectiveness of the suggestions was measured by considering the differences between two ratings: the first was based on the explanation and the second was the one given after trying the item. This approach has been also considered in [42] in order to measure the effects of personalization in RS. These proposals allow users to judge the recommendation according to the explanation provided, requiring a post-visualization evaluation. The influence of the selected interface in the user's feedback was analyzed in [6] and more recently in [9]. Correspondingly, Pu and Chen [30] show that displaying a diverse set of results on an organization-based interface is more effective and enables user trust formation to be compared to the simple k-best interface even after the "why" enhancement. A similar conclusion was reached by Hernando et al. [18,28] who used a tree to display an overall explanation of all the recommendations instead of providing an independent explanation for each recommendation. In the tree, the items are related by taking into account the similarities between their ratings, helping users to decide which recommendation to choose. A different perspective is to use external informational elements in the explanations, as in the case of those that can be

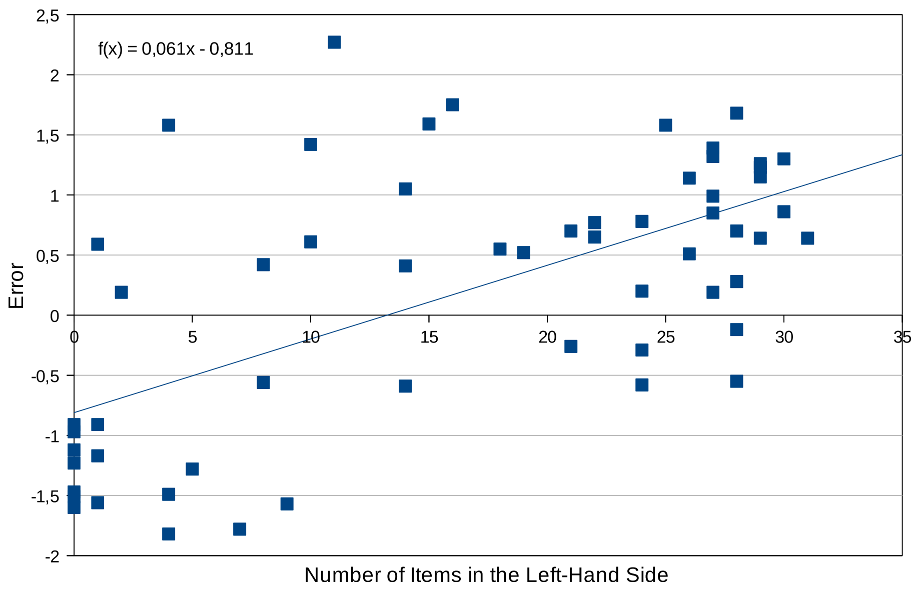

Fig. 3. Correlating the error in the predictions with the support (number of items) in a 2D explanation.

obtained from social networks [35], the content description associated with products [34] or location-based information [39].

Another user study evaluated the effects of certainty rating on perceived transparency, and acceptance of the system and its recommendations. In this study, these explanations did not cause an increase in any of these factors. However, the authors also stated that these explanations were not always understood by users and that confidence-based explanations may be better for improving the trustworthiness of a recommender system rather than its transparency or persuasive capabilities [7].

There are also various case-based lines of research into scrutability which are connected with our proposal. In these studies, explanations are used to correct the reasoning of the RS. From the explanations, the user can understand any misguided assumption, changing the type of recommendation they receive or refining their profile [37,41]. A case-based RS to explain the relevance of any question users are asked in terms of their ability to discriminate between competing cases is presented in [26].

In all of the previous studies, users played an active role in the processes: users observed an isolated explanation and some of their actions were consequently analyzed. Although similar items can be used as part of an explanation, the lesson learned from the explanation for these items is not normally used, except in those cases where users modify their profiles. In our approach, we propose to learn without explicit user involvement and then use these lessons in order to improve the recommendations.

It has been found that bringing low quality recommendations to the attention of experienced users can decrease their satisfaction with the system, most likely because it alerts these users to the fact that the system might be wrong [25]. In this respect [38], identify that it is possible for a trade-off to occur between the accuracy of a recommendation and the quality of its explanation. In other words, a recommendation may sometimes be accurate but may not be well justified and vice versa. In published work, unconfident recommendations are usually associated with those situations where there is little data to support them [24,25].

In order to solve this problem, and closely connected with our approach, in [5] we use data gathered from histogram-based explanations to determine the reliability of a prediction. In this way, we provide the system with the ability to detect high confidence recommendations, and use explanations that pinpoint low confidence sparingly. In order to determine confidence, we have considered both (a) the predicted rating for an item and how it was computed (entropy) and (b) the errors found when analyzing recommendations for similar already-rated items. Therefore, if predictions were not reliable in similar cases (i.e. large errors were obtained), we would be wary of this recommendation.

From this work, we have learned that some knowledge can be learned from a past-based explanation, i.e. an explanation that includes information of the behavior of the system over a set of observed items. This is the information that we want to exploit in this paper in order to improve the system's prediction. With the elements we have discussed so far, we have found no evidence of any approach that employs a similar mechanism for recommending.

Following [29], there are three fundamental resources that can be used in an explanation: (i) the Human style of explanation, which provides explanations based on similar users; (ii) the Item style of explanation, which is based on choices made by a user on similar items; and (iii) the Feature style of explanation, which explains the recommendation based on features of an item that were rated by the user beforehand. Our approach can be categorized as a hybridization of the Human and Item style.

4. Learning from explanation

In our approach, we shall attempt to learn using information gathered from a set of explanations. The given explanation, however, strongly depends on the recommendation model used. Several recommending strategies have been published [33], but in this paper we shall focus on a nearest-neighborhood-based collaborative filtering algorithm which computes the predictions by considering how similar users rated a target item. As mentioned previously, not only do we have an RS that computes a prediction for a target item $\hat{r}(a,t)$ but also an explanation interface (see Section 2) which includes both a histogram showing information about how this prediction was computed and a 2D graph which includes a set of predictions for a subset of items (the explanation support $\mathcal{S}$) rated by the active user.

After observing the explanations, users can assess the quality of the prediction and, if any doubt remains, reconsider the predicted value. We believe that the process of redefining the predicted rating after observing an explanation can be automated. In order to do this, we propose that a set of numerical data be gathered from the explanations. These data will then be used as the input for a machine-learning algorithm in order to learn the error in the prediction (e.g. we might learn that the system tends to underestimate the user's ratings). This process shall be transparent to the user, i.e. it is a pre-visualization approach.

In order to achieve our objective, two main components must be studied:

- Firstly, we need a module that is capable of analyzing the explanation (both the histogram and the 2D graph). In Fig. 4 we present a graphical description of our approach. More specifically, given a graphical explanation for the target item I_t (bottom-right of the graph), we create a tuple (a row) containing a set of numerical features summarizing the information presented in the explanation, i.e. $\mathcal{T}_t = \{f_{t,1},\ldots,f_{t,s}\}$. Some of these features are predicted rating, number of items support-

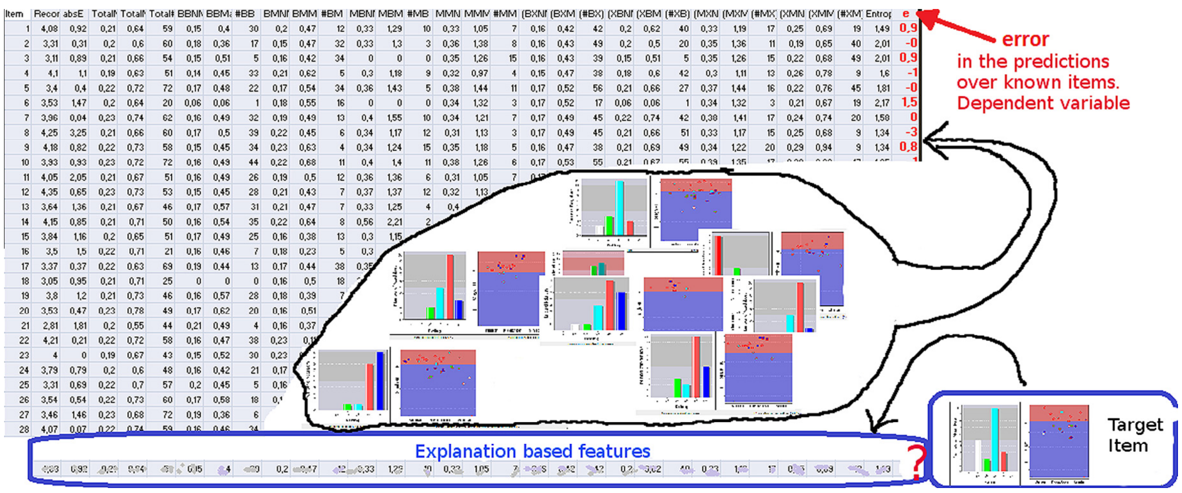

Fig. 4. Gathering data from the histogram and the 2D-based explanation interfaces.

ing the recommendations, entropy, etc. (see Section 4.1). In a subsequent second step, we then compute a prediction and its associated explanation for each item rated by the active user, $I_o \in \mathcal{R}$. From these explanations we extract the same numerical features as before (rows in Fig. 4) plus an additional feature that represents the user action, computed as the prediction error, $e_o = r_o - \hat{r}_o$ (last column of the rows (in red[2])). For each observed item, I_o, the set of features is therefore $\mathcal{T}_o = \{f_{o,1}, \ldots, f_{o,s}, e_o\}$.

For the sake of simplicity, we divide the features into two subsets which are formed from how the target item's prediction was computed by the RS, the so-called RS-based features, $F_{RS}(t)$, and from the performance of the system for certain known (rated) items $\mathcal{S}$, the so-called explanation-based features $F_{Ex}(t)$. In order to illustrate our approach, we shall consider that $F_{RS}(t)$ only includes the predicted rating $\hat{r}(a,t)$ and that $F_{Ex}(t)$ is the average error on the predictions for the items supporting the explanation $\mathcal{S}$.

- Once these data have been obtained, we can use an inductive learning approach that will, from a set of training examples with a known error, i.e. $\mathcal{T} = \{\mathcal{T}_o, \forall I_o \in \mathcal{R}\}$ ($\mathcal{T}$ represents the specific explanation-based knowledge), be capable of discovering a function which is able to estimate the prediction error for the target item, $\hat{e}_t$, given the values obtained from the explanation for the target item, i.e. $\hat{e}(t) = f(F_{Ex}(t), F_{RS}(t))$. The prediction error therefore acts as the dependent variable whereas the set of features are the independent variables. If required, the estimated error can then be used by the action module to modify the recommendation, i.e. the new recommended value can be computed as $\hat{r}'(a,t) = \hat{r}(a,t) - \hat{e}(t)$.

  Following on with our simplified example, what we are looking for is a set of rules which express, for example, that if the predicted rating is 2 and the average error over previous predictions is 1.5 then the expected error in the prediction is $\hat{e}(t) = f(2, 1.5) = 0.7$.

  More specifically, in this paper we propose the use of the $M5'$ algorithm (see Section 4.2 for more details), which builds a binary tree classifier, where each leaf can predict the error using linear regression. Predictions are computed by tracing the path to a leaf, using binary tests for a single feature.

4.1. Selected features

In this section, we shall describe the set of features selected by our approach in order to obtain the different instances necessary for the learning process. Let us assume that we are viewing the explanation for an observed item, I_o. We can therefore obtain the following set of features:

- F_{RS}: directly related to the observed item and obtained from the RS:
  - $\hat{r}(a,o)$, the recommendation or predicted rating.
  - $e(o)$, the error in the prediction (class attribute), defined as $e(o) = r(a,o) - \hat{r}(a,o)$. It should be noted that this is the error that we are trying to learn for the target item.
  - $H(o)$, the entropy that measures the variety (uncertainty) in the distribution of the ratings given by the selected neighbors to the item, defined as

$$H(o) = \sum_j p(r_j) \log_2(1/p(r_j)),$$

[2] For interpretation of color in Fig. 4, the reader is referred to the web version of this article.

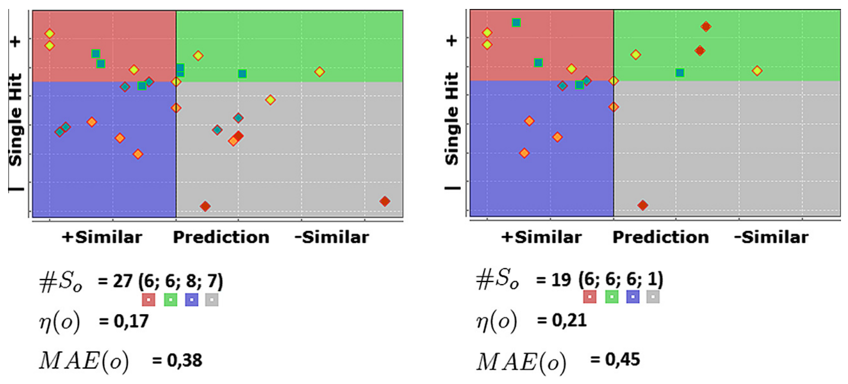

Fig. 5. Splitting the 2D graph into four parts considering similarity and hits.

where the sum is over the set of alternative ratings (e.g. 1★ to 5★), representing $p(r_j)$ the probability that the neighborhood rated the item with the value r_j.

- F_{Ex}: extracted from the set of items that should be presented in the 2D graph-based explanation, $\mathcal{S}_o$.
  - $\#S_o$, the number of items on the graph (explanation support).
  - $\eta(o)$ that averages the individual hits (see Eq. (3)) among the set of items in $\mathcal{S}_o$,

$$\eta(o) = \frac{1}{\#S_o} \sum_{i \in \mathcal{S}_o} h(i).$$

  - $MAE(o)$ represents the mean absolute error in the predictions for those items in S_o, i.e.

$$MAE(o) = \frac{1}{\#S_o} \sum_{i \in \mathcal{S}_o} |\hat{r}(a,i) - r(a,i)|.$$

The difference with $\eta(o)$ is that when the MAE is considered, the different ratings are combined (as a weighted average) to determine the prediction for the item.

We have also observed how the distribution of the items on the 2D graph might be relevant for determining the quality of the prediction. We have therefore divided the grid into four parts[3] (see Fig. 5). Not only have we considered the similarity between predictions by using a threshold of 0.25 to split the X-axis and dividing the graph into the left-hand (most similar) and right-hand (least similar) sides but we have also considered those items for which the system can properly predict using a threshold of 0.2 for the hit measure, Y-axis, by splitting the graph into the upper part (good individual predictions) and lower part (wrong individual predictions).

Explanation-based features will also be computed for the different parts of the grid. In order to distinguish each part, we shall use a tuple $-sh-$ as a prefix, where the first component, s, takes its values in $\{l, r, x\}$ where l represents the left-hand side, r the right-hand side and x is used to represent both. Similarly, the h component takes its values in $\{t, b, x\}$, representing the upper part, lower part and both parts, respectively. Thus, for instance, $lt\_\eta(o)$ represents the η metric computed over those items in the upper left-hand corner of the 2D graph (the most similar and best predicted items), $xb\_\eta(o)$ represents this metric for the lower part, and $xx\_\eta(o)$ is used to denote the metric for the entire graph.

Using this approach, we shall obtain a set of 30 different features for each observed item. For the target item we can obviously compute every feature except the prediction error which is the one we want to determine. For example, for the 2D-explanation for the film "Schindler's list" (represented on the left-hand side of Fig. 5), we are able to compute the following values simply by counting the number of films in each part of the graph: $lt\_\#S_o = 6$, $lb\_\#S_o = 8$, $rt\_\#S_o = 6$, $rb\_\#S_o = 7$, $lx\_\#S_o = 14$, $rx\_\#S_o = 13$, $xt\_\#S_o = 12$, $xb\_\#S_o = 15$ and $xx\_\#S_o = 27$; $xx\_\eta(o) = 0.17$ and $xx\_MAE(o) = 0.38$.

4.2. M5: Foundations and application to the problem

One of the constituent parts of the area of machine learning [13] is inductive learning, or learning by example. In this field, the algorithm attempts to induce a general rule from a set of observed instances. The paradigm of this type of technique is classification, which basically consists in assigning to a given instance the class to which it belongs.

There is a wide range of approaches for such a task and of all of these, we should mention the family of decision trees. A classifier is built in the form of a tree, where each node represents an attribute of the problem. For root and internal nodes, tests of their corresponding attributes are conducted for a given instance, so each descending branch is assigned to one of the possible values for the attribute. Finally, leaf nodes offer a value for the class attribute. Decision trees are also expressed in

[3] It should be noted that we did not show this division to the users; it is merely for analytical purposes.

the form of rules. The rule base comprises as many rules as leaf nodes on the tree. The antecedent is formed by the combination of all the tests from the root node to the node before the leaf. The value that the class variable will take appears in the consequent.

Decision trees are efficient, robust and relatively simple classification techniques. One of their main additional advantages from the human point of view is that they are understandable. They are divided into two groups: classification trees, where the class is a discrete variable, and regression trees, where this attribute is continuous. Leaf nodes then store a kind of average for the class attribute for all the instances that reach this leaf node. Examples of such classifiers are the classic ID3 and C4.5 algorithms [31]. If leaves store linear regression functions that predict the class value, the tree is instead called a model tree. In such a case, we should mention M5 [31] as this is one of the classic algorithms and the one we have selected to put our proposal into practice, although we have actually used the $M5'$ implementation [43], which has been improved in various ways (management of discrete variables and missing values and reduction of the tree size), contained in the Weka [12,44] data mining toolkit.[4]

Following Quinlan's own description, the M5 algorithm basically comprises the following steps, assuming a collection of T training instances:

1. Construction of a decision binary tree following a divide-and-conquer approach. The training instances are partitioned until the examples in the leaves form relatively homogeneous sets. The splitting criterion is to choose the attribute that minimizes the intra-subset variation in the class values for each value of this attribute, i.e. the standard deviation of the class values that reach a node is computed, $sd(T)$; this same measure is then computed for each set T_i, corresponding to the values that this attribute could take, $sd(T_i)$. Finally, the node that maximizes the expected reduction in error (the difference between $sd(T)$ and $\sum_i sd(Ti)$) is selected. This process is repeated recursively for each remaining subset.
2. Construction of a linear model in each node of the tree but considering only the nodes contained in the corresponding subtree.
3. Computation of the estimated error in each node for unseen instances.
4. Simplification of linear models in each node, removing variables that poorly contribute to the model.
5. Pruning the tree from leaves to the root. Nodes will be sequentially removed while the estimated error decreases (comparing the node error with the one obtained in their subtrees).
6. Smoothing process to improve the prediction accuracy of the models and compensate discontinuities that could occur in adjacent models in the leaf nodes.

In the Weka $M5'$ implementation, the following main parameters may be controlled: minimum number of instances to consider splitting, use of unpruned tree/rules and use of unsmoothed predictions.

By way of example, in the context of this paper, given a set of training instances for the MovieLens user with ID 507, who rated 58 items, a tree obtained as the $M5'$ output from the Weka implementation is shown in Fig. 6. This tree is obtained by considering the explanation given for the movie "Everyone Says I Love You" and contains a total of four variables represented as internal nodes. Each edge of the tree is labeled with the corresponding attribute value. The leaves, which correspond to the class variable, error e, or in terms of regression, the variable with which we would like to predict a value, are labeled with the identifier of the linear model (LMi) learned, the number of instances that fit the corresponding leaf node (a kind of rule support, i.e. the number of instances so that each test holds true from the root to the leaf) and the error gain percentage, δ, in this node in relation to the instances supporting the rule.

The error gain percentage obtained in the leaf node is the root mean square error (RMSE) of the instances in the node divided by the global standard deviation of the class. In those cases where the percentage value is lower than 100%, the estimation is therefore doing a good job.

In Fig. 6, we can observe five rules (one for each leaf) comprising the combinations of the tests from the root to a leaf node. For each rule, the learned linear models are:

```
LM1: e = 0.0205 * rx_#S − 0.3349
LM2: e = 0.0205 * rx_#S − 0.2952
LM3: e = 0.0249 * rx_#S − 0.1412
LM4: e = 0.0072 * rt_#S + 0.0103 * rx_#S − 0.2176
LM5: e = 0.0050 * rt_#S + 0.0103 * rx_#S − 0.2194
```

How can this tree model be used to redefine the recommendation for a user? When a new instance from a user arrives, the various branches are basically tested in relation to the attribute values, reaching a leaf node. At this moment, the value of the variable error is predicted using the corresponding linear model. The predicted rating is then re-computed.

We would like to discuss the use of pruned or unpruned trees in $M5'$, i.e. the use of a pruned criterion might give simple rules (even with zero or one variable). This is because the overall error is reduced after pruning the branch of a node. This situation usually works well when a prediction is necessary for any instance, but this is not our case. We already have a rec-

[4] http://www.cs.waikato.ac.nz/ml/weka/.

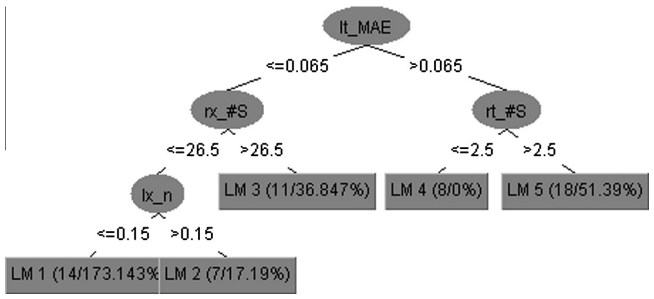

Fig. 6. *M5′* Model tree obtained by considering the explanation given for the prediction of the movie “Everyone Says I Love You” to User 507 in the MovieLens data set.

ommendation, and we can choose whether to act on this recommendation or not. By using an unpruned strategy, we can therefore preserve the good branches, although the learned model might overfit the data (in the worst case obtaining one rule for each instance). In order to avoid overfitting, we require a large number of instances to split a node, denoted as $\#I$. Weka uses a default value of 4 but we shall consider greater values (see Section 5.1 for a discussion).

It is not always possible, however, for the application of a model rule to improve the predictions and this means that our confidence in the explanation is not necessarily well justified. When the model rule is in fact applied to every instance on a massive scale, MAE decrease is extremely probable. Two factors in the tree leaves will determine rule application:

- The first concerns the error gain percentage, δ, which will have a significant impact on the results (see Section 5.1) verifying that the greater the value of δ, the lower the accuracy of the rule. In order to illustrate this, we shall again focus on User 507 considering the MAE for all their predictions[5] as the accuracy metric. For this user, the MAE value for the original RS is 0.564 and we can reduce its value to 0.531 by applying the linear model for all the instances, i.e. for each of the 58 predictions the corresponding rule was triggered. Taking into account a percentage $\delta \leqslant 70\%$ only 25 rules are triggered (25 predictions were redefined), decreasing the MAE value to 0.485. More significant results are obtained with a percentage $\delta \leqslant 50\%$, with an MAE value of 0.477 with only 14 instances (rules). However, by applying the rules with a percentage $\delta \leqslant 40\%$ the MAE increases to 0.482 (10 rules were fired).
- The second concerns the number of instances supporting the rule. If this value is low, then our confidence in terms of the error gain percentage is also low, as there are few instances stored in the corresponding leaf node. We set the minimum number of required instances to 10.

Finally, we should mention that we could use any other regression model that learns a decision tree, and not necessarily *M5′*. We have used this algorithm because it is a well-known, state-of-the-art regression algorithm, which first outputs a decision tree and then the rules. Any algorithm fulfilling these restrictions could be used in our model to learn explanations.

4.2.1. Example

In this section, we shall present an example in order to clarify our purposes as well as the foundations for our proposed approach.

Table 1 shows all the features extracted from the explanations for User ID 1 in the MovieLens data set, target item I_t = “Monty Python’s Life of Brian (1979)”, and six different observed items. Columns $O1, O2, \ldots$ of this table represent the training instances for the learning algorithm, which comprise a total of 30 features most of which are obtained from the explanation interfaces. In the case of the target item, we do not know the last variable, *Estimated Error*, which is represented by a question mark in Table 1. This is the dependent variable that we wish to learn by means of the regression algorithm.

When feature computations, and the corresponding extraction, are performed for a given user and item obtained from the graphic explanation, the new instance is consequently sent to the regression module, which computes an estimation of the error. More specifically, the features involved are highlighted in Table 1 and the triggered rule is:

$$\text{if } ((H(t) > 1.3) \text{ and } (lb_{MAE(t)} > 1.32) \text{ and } (lx_{\#S(t)} > 30)) \text{ then } \hat{e}(t) = 0.0408 * lb_{MAE(t)} + 0.0182 * lt_{\#S(t)} - 0.151.$$

With this new value, a new rating is computed by adding the error to the original predicted rating, in this way simulating the process that the user carries out when shown the explanation.

In Table 1, the RS originally predicted a rating of 3.94 for the movie “Monty Python’s Life of Brian”. The regression algorithm estimated an error of 0.40, so a new 4.34 prediction was generated, considering that the first recommendation was an underestimation, and this was relayed to the user (it should be noted that the user rated this item with a 5).

[5] The performance is measured using the leave-one-out methodology, learning a different model tree for each target item.

Table 1
Some instances for learning, expected error and modified predicted rating.

Features	Target	Already rated items						
	I_t	O1	O2	O3	O4	O5	O6	...
$\#S_o$	53	64	36	31	27	60	49	...
$MAE(o)$	0.7	0.64	1.02	0.72	0.78	0.73	0.68	...
$\eta(o)$	0.22	0.21	0.28	0.21	0.22	0.22	0.22	...
$lt_{\eta(o)}$	0.15	0.17	0.17	0.15	0.17	0.11	0.18	...
$lt_{MAE(o)}$	0.43	0.46	0.55	0.71	0.61	0.39	0.55	...
$lt_{\#S_o}$	**26**	37	14	2	17	3	15	...
$rt_{\eta(o)}$	0.2	0.18	0.19	0.18	0.19	0.18	0.17	...
$rt_{MAE(o)}$	0.59	0.42	0.63	0.61	1.01	0.52	0.36	...
$rt_{\#S_o}$	14	13	7	25	3	42	11	...
$lb_{\eta(o)}$	0.46	0.35	0.47	0.32	0.34	0.38	0.39	...
$lb_{MAE(o)}$	**1.83**	1.37	1.93	0.85	1.12	1.44	1.43	...
$lb_{\#S_o}$	6	13	3	1	5	4	9	
$rb_{\eta(o)}$	0.32	0.48	0.41	0.45	0.44	0.34	0.31	...
$rb_{MAE(o)}$	0.99	1.87	1.57	1.66	1.09	1.33	1.1	...
$rb_{\#S_o}$	7	1	12	3	2	11	4	...
$xt_{\eta(o)}$	0.17	0.17	0.17	0.17	0.17	0.17	0.18	...
$xt_{MAE(o)}$	0.48	0.45	0.58	0.62	0.67	0.52	0.45	...
$xt_{\#S_o}$	40	50	21	27	20	45	21	...
$lx_{\eta(o)}$	0.21	0.22	0.22	0.21	0.21	0.26	0.24	...
$lx_{MAE(o)}$	0.69	0.67	0.79	0.76	0.73	0.99	0.8	...
$lx_{\#S_o}$	**32**	50	17	3	22	7	24	...
$xb_{\eta(o)}$	0.38	0.36	0.42	0.37	0.35	0.35	0.35	...
$xb_{MAE(o)}$	1.38	1.32	1.64	1.45	1.11	1.36	1.27	...
$xb_{\#S_o}$	13	14	15	4	7	15	8	...
$rx_{\eta(o)}$	0.24	0.2	0.33	0.21	0.29	0.21	0.21	...
$rx_{MAE(o)}$	0.73	0.52	1.22	0.72	0.69	0.56	0.52	...
$rx_{\#S_o}$	21	14	19	28	5	53	15	...
$\hat{r}(a,o)$	3.94	3.91	4.14	1.73	3.08	2.96	2.96	...
$H(o)$	**1.85**	1.56	0.78	0.53	1.6	2	1.91	...
$e(o)$	?	1.09	−0.14	0.27	0.92	−0.96	1.04	...
Estimated error	0.40							
New rating	4.34							

4.3. *Efficiency considerations*

We wish to conclude this section by discussing the costs of our approach. Correspondingly, we must compute various explanations: one relating to the target item and others relating to previous items. Constructing such explanations is not hard due to the low number of items rated by the user. Moreover, explanations for previous items can be precomputed and, once the model has been learned, the computation of the final prediction is not affected. Nevertheless, there might be a high computational cost when the problem is to compute a list of recommended items, instead of the rating's prediction. In this situation, this process will only be executed for those items at the top of the ranking in the primary list.

Finally, we would like to say that although our approach is designed for neighborhood-based RS, it could be applied with other recommendation strategies, for instance to learn the importance of certain features in a content-based RS. Accordingly, our approach can be used to learn from previous experience, the features of which are used by users to make decisions. However, we believe that this migration should be undertaken with caution.

5. Evaluation

The final objective is to show whether it is possible to determine the error in a recommendation by analyzing data gathered from an explanation. This will help us conclude that explanations can be considered as a valuable source of knowledge that might be exploited in the recommending processes. In order to tackle this objective, we shall attempt to answer the following research questions:

Q1: *Is our approach a one-size-fits-all solution?* When designing our explanation interface, we realized that users' tastes vary considerably. For some users, an explanation contains enough information while for others it is lacking. Similar results have been previously obtained in other research [42], concluding that it is quite difficult to understand/measure the effects of explanation in RS, probably because the same explanation is not appropriate for each system user. In Section 5.2, we shall therefore see that the effects of an explanation are user dependent. We consider that finding those users for which a particular explanation interface might be beneficial is highly relevant for the RS community, and further developments in this direction will be needed.

Q2: The next question we tackle is *What benefits will there be after applying our approach?* Although there are several subjective criteria that could be taken into account, we shall focus on the reduction of the expected error in the recommendation, i.e. to determine whether we can obtain more accurate recommendations or not. In this case, see Section 5.3, we will see that when we require greater confidence, the system will trigger a low number of high quality rules. Otherwise, we can trigger a large number of rules, but the performance might decrease. A balance between both situations should be achieved.

Q3: Finally, Section 5.4 considers whether *certain attributes are more relevant for predicting the error in the recommendations*. By means of this analysis, we shall gain a better understanding of the processes behind our 2D-based explanation and the role of the different attributes under the performance metrics.

It is evident that any of the previous research questions are related to the regression algorithm that is used to learn from explanations. We believe that the main focus of the paper is to determine whether the general idea of learning from explanations may work. For this reason, $M5'$ is therefore only a tool that we apply and not a target itself.

5.1. Experimental framework

In this section, we shall describe the general experimental framework on which we will base our experimentation in order to answer the previous questions. Generally speaking, we would require a test data set and CF algorithms to perform recommendations on it; RS evaluation metrics to validate the accuracy of the recommendations; and the regression algorithm, with its respective parameter assignments and an evaluation method:

- *Dataset:* Since our approach does not require any user interaction, we have decided to validate it using an empirical evaluation based on the classic MovieLens data set [27], which was collected by the GroupLens Research Project at the University of Minnesota containing 100,000 anonymous ratings (on a scale of 1–5) of approximately 1682 movies rated by 943 MovieLens users who joined MovieLens during the seven-month period between September 19th, 1997 and April 22nd, 1998.
- *Underlying CF approaches:* In order to validate our proposal with different models, we have used two collaborative filtering approaches in the experimentation, which we shall briefly describe, according to the two main CF stages (neighbor selection and prediction):

  PCCF: In this state-of-the-art memory-based collaborative filtering model, as proposed by Herlocker et al. [16], the best neighbors for recommending the target item, $N_t(a)$, are selected using Pearson's Correlation Coefficient (PC) among the individuals who rated it, devaluing correlations that are based on small numbers of co-rated items, k. The similarity between two users is therefore computed as follows:

$$sim(U,V) = PC(U,V)\cdot CF, \quad (4)$$
$$\text{with } CF = \begin{cases} 1 & \text{if k } > 50 \\ \frac{k}{50} & \text{otherwise} \end{cases}$$

In terms of the prediction phase, the predicted rating for the active user, $\hat{r}_{a,t}$, is computed using the following classic equation in the RS field:

$$\hat{r}_{a,t} = \bar{r}_a + \frac{\sum_{V\in N_t(a)}(r_{v,t} - \bar{r}_v)sim(A,V)}{\sum_{V\in N_t(a)}sim(A,V)} \quad (5)$$

  where $\bar{r}_u$ represents the mean rating for user U and $r_{u,i}$ represents the true rating given by User U to Item I_i. In this case, it is assumed that users may use a different rating scale to quantify the same level of preferences for an item.

  ELCF: This is based on predictive capabilities instead of rating similarities, i.e. a previous prediction-based CF [3,19]. This algorithm also follows a neighborhood-based approach by selecting as neighbors those users which best predict user ratings. The idea is that if a user is good at predicting the active user's previous ratings, then they will also be good at predicting unobserved ratings. More specifically, User U will be included in the active User A's neighborhood if their expected loss over the active users previous ratings, R_A, is high enough and this is measured as follows:

$$EL(A,U) = \frac{\sum_{i\in R_A} L(r_{a,i}, \hat{r}_{a,i}(u))}{|R_A|}, \quad (6)$$

  where L is the loss defined by $L(r_{a,i}, \hat{r}_{a,i}(u)) = abs(r_{a,i} - \hat{r}_{a,i}(u))$, i.e. the average absolute deviation between the user's true rating, $r_{a,i}$, and the predicted rating, $\hat{r}_{a,i}(u)$ defined as $\hat{r}_{a,i}(u) = \bar{r}_a + (r_{u,i} - \bar{r}_u)$. The same CF as in Eq. (4) is used to penalize a low number of co-rated items.

  The prediction stage is performed using a mean-centering suggestion, as in Eq. (5), but taking into account the already mentioned expected loss in such a way that greater weights are given to those users with more accurate predictions.

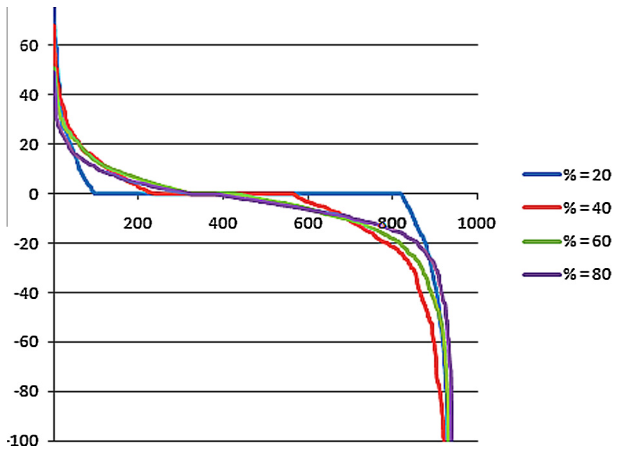

Fig. 7. MAE improvements (*Y*-axis) for each one of the different users (*X*-axis) after using explanation-based knowledge. The different lines represent the results obtained when varying the δ threshold for trigger a rule, being #I = 20.

- *Recommendation evaluation metrics:* In order to test performance, we shall consider system accuracy using two standard metrics [11,17]: the MAE (mean average error) and the RMSE (root mean squared error), which basically differ in the loss associated to each error. The RMSE therefore gives a relatively greater weight to large errors and is most useful when large errors are particularly undesirable.
- *Regression algorithm:* As we have already mentioned in several sections of this paper, the prediction error will be estimated by means of a regression algorithm, and more specifically, the state-of-the-art $M5'$ algorithm which fulfills our purposes and performs reasonably well. With respect to the parameters of $M5'$, we shall use an unpruned strategy requiring a minimum number of instances ($\#I$) to split a node. When analyzing the resulting model tree, we then consider two restrictions to trigger a rule:
  1. The number of cases in the leaf (rule support) must be $\geqslant$ 10 and
  2. The error gain percentage must be less than a threshold, represented as δ in this experimentation.
- *Learning evaluation:* in order to validate our model, whenever the RS computes a prediction for the active user, we must learn a decision tree from their previous experiences. Like most machine-learning evaluation methodologies, and because the user does not usually rate a large number of items, we have applied a leave-one-out strategy.

5.2. Q1: Is our approach a one-size-fits-all solution?

In a first experiment, we apply our proposal to all the users in the MovieLens data set using the ELCF model by considering a different number of instances to split a node ($\#I = 10,$ 15 and 25). Focusing on $\#I = 20$, for example (although the trends are similar for each case), we can see that the overall MAE (0.718) is reduced to $0.706,$ $0.708,$ 0.713 and 0.717 by using a δ threshold which is equal to $80,$ $60,$ 40 and 20, respectively, to trigger the rules. These results might be considered disappointing: the computational efforts have minor effects on the accuracy of the model (improvements of around 1.6% in terms of the MAE).

When we focused on the data, however, we observed that performance differs considerably between users, as Fig. 7 illustrates. The *X*-axis represents the 943 users sorted according to improvements in the expected error, which is represented on the *Y*-axis. We also show the results of considering different values for the δ threshold to determine the triggered rules, ranging from 20% to 80%. Generally speaking, if we look at the data, we can detect a group of users for whom the explanations help correct predictions and other users where our approach worsens the predictions given by the RS (those on the left-hand side and right-hand side of Fig. 7, respectively). This shows that explanations are not a one-size-fits-all solution, which might explain why it is difficult to measure their effects [42].

What is important is that this performance is highly user-dependent, i.e. there is a subset of users which enable accurate learning from the explanations, and these can therefore be detected in advance (from their previous experiences). Our first objective, therefore, was to identify these users. More specifically, and with this objective in mind, we use a subset of already rated items as the validation set and subsequently, if after applying our proposal for these items, the performance of the system decreases (we obtain a greater MAE after adjusting the recommended ratings), the user was excluded from our learning strategy, i.e. for these users we shall always recommend the original predicted rating.[6]

As a result, we restrict our methodology to a subset of users. It is important to note that both the number of selected users and also the number of items for which a rule is finally triggered depend on two parameters: the δ threshold and the

[6] In a future study, we plan to focus our research on these users.

Table 2
Number of users (U), rated movies (M) and triggered rules (R) with the ELCF model.

δ	#I = 15			#I = 20			#I = 25		
	U	M	R	U	M	R	U	M	R
10	48	11,439	419	34	7604	319	26	5850	237
20	142	30,223	1340	101	20,940	1088	75	16,663	781
30	209	37,607	2400	168	32,024	2109	155	28,962	1684
40	263	39,034	4307	244	38,969	4227	225	36,687	3680
50	313	41,964	7576	289	44,090	8533	282	44,189	7832
60	329	41,616	11,091	326	45,018	13,489	312	46,070	13,013
70	350	44,383	15,211	334	43,839	18,022	306	44,074	17,982
80	337	41,079	16,934	340	44,499	22,690	311	43,998	23,382
90	344	39,989	18,623	334	41,174	24,654	311	41,499	26,475

parameter #I. In Tables 2 and 3 we show how these are related to the different values of the parameters for both the ELCF and the PCCF RS. For each group, the first column shows the number of users (U), the second the number of movies (M) that they rated in the original MovieLens data set and then the numbers of triggered rules (R), i.e. items with revised recommendations. From these data, we can conclude[7] that the δ threshold is important: if we require high confidence in the rules, a low number of rules will be triggered. Being less restrictive will enable more rules to be triggered and also more users to be selected.

Nevertheless, the number of users with triggered rules stabilized at around 25–35% of users in the original MovieLens data set with δ values greater than 50%. We might conclude that good rules seem to focus on a low number of users. Increasing δ will allow more rules to be triggered but to the same users. There is no considerable difference in the results, on the other hand, in terms of #I. By using large values of #I, high level rules are obtained that can be applied more widely but with no great change in the number of users. Once again, this seems to suggest that our approach is user-dependent.

5.3. Q2: Measuring the error gain

We consider the above results to be valuable because the system could improve the results for quite an important percentage of customers, whereas the other users will receive the same recommendations. For example, if #I = 20 and $\delta = 50$, the MAE for the 289 users and 44,090 ratings using ELCF is 0.7181 whereas if we update the 8533 recommendations, this value can be reduced to 0.6997. However, this second MAE value highly depends on the number of items when the system does not trigger a rule, overshadowing the importance of the improvements. In order to focus on the benefits of our approach whenever it is applied, the performance values presented in this section have therefore been obtained by considering only those situations where a rule has been triggered.

Tables 4 and 5 present the results obtained when considering a different number of items in the splitting strategy ($\#I$) and different values of the δ threshold for the ELCF and PCCF models, respectively. We show the MAE obtained by the original model, the MAE obtained using our approach ($M5'$) and the gain improvements in terms of percentage (%G). Various conclusions can be drawn from these tables. There is no clear winner in terms of $\#I$ but in the case of the δ threshold, we find that as its value increases so does the error: we trigger rules of a lower quality yet increase the number of rules and the number of users who can equally benefit from our approach, as discussed in Section 5.2. A balance between these two opposing criteria must be achieved.

In order to further analyze this situation, we shall focus on the data in Fig. 8 that instead of presenting absolute values considers the relative ones for #I = 20 (the remaining values exhibit similar trends). On the X-axis we present the error gain, δ, whereas on the Y-axis we present the relative percentage for the different performance metrics for the two RSs. In particular, we show the relative percentage of users, percentage of rules (in terms of the items rated by the users, denoted by Items(U), and also the total number of items, denoted by Items) and percentage of MAE improvements, denoted by %G. In this case, we can see that by using lower δ values very good rules can be obtained although these can only be applied a few times. Increasing this value to around 50–60% allows us to apply the model to a large number of users (30–40%) in the original MovieLens data set. For these users, a rule will be triggered for around 20–30% of the recommendations, resulting in valuable improvements of around 10–15% (in terms of the MAE). Although an increase in threshold will not result in a considerable increase in the number of users, a large number of rules will, however, be triggered leading to smaller improvements (lower than 10%). We therefore believe that a good balance is to consider a threshold of around 50–60%. Nevertheless, better results will be obtained if we are able to identify the best threshold for each user.

Finally, Tables 6 and 7 show the values of the RMSE metric for the different values of the threshold for the ELCF and PCCF models, respectively. The performance is similar to that obtained with the MAE values for the first, although the improvements are less consistent. In our opinion, this is due to the fact that RMSE penalizes larger errors, and these are more difficult to discover with $M5'$. We know that this algorithm is closely tied to the characteristics of the data set. In our case, there is not a huge percentage of large errors in the original MovieLens data set. More specifically, the number of predictions with an

[7] We shall analyze the accuracy of our approach in the next section.

Table 3
Number of users (U), rated movies (M) and triggered rules (R) with the PCCF model.

δ	#I = 15			#I = 20			#I = 25		
	U	M	R	U	M	R	U	M	R
10	74	15,287	861	64	12,652	777	53	10,402	421
20	174	32,208	1948	142	27,089	1718	127	23,244	1307
30	255	42,018	3267	207	36,855	2909	175	30,257	2304
40	335	47,259	5377	297	45,472	5283	262	41,266	4451
50	389	50,078	8778	365	48,971	9029	322	46,921	8073
60	414	50,835	12,908	387	48,719	13,940	344	47,012	13,171
70	421	47,359	15,986	401	48,506	19,401	365	48,579	19,650
80	440	49,311	20,028	425	49,329	24,692	385	47,905	24,940
90	439	47,079	21,880	412	46,980	27,740	377	46,613	29,196

Table 4
MAE values using the original predictions for the ELCF (EL) model and those obtained by redefinition having learned the error with $M5'$, column $\%G$ represents the gain percentage.

δ	#I = 15			#I = 20			#I = 25		
	EL	$M5'$	%G	EL	$M5'$	%G	EL	$M5'$	%G
10	0.532	0.398	25.1	0.464	0.342	26.2	0.493	0.390	20.9
20	0.587	0.454	22.5	0.584	0.440	24.6	0.605	0.440	27.2
30	0.624	0.491	21.2	0.620	0.481	22.4	0.605	0.475	21.5
40	0.683	0.558	18.2	0.684	0.558	18.3	0.669	0.535	20.1
50	0.690	0.594	14.0	0.688	0.593	13.8	0.679	0.582	14.3
60	0.708	0.626	11.6	0.699	0.621	11.1	0.695	0.619	10.9
70	0.721	0.653	9.4	0.718	0.652	9.1	0.699	0.637	8.8
80	0.725	0.661	8.7	0.723	0.669	7.5	0.713	0.661	7.2
90	0.729	0.669	8.2	0.721	0.666	7.6	0.709	0.659	7.0

Table 5
MAE values using the original predictions for the PCCF (PC) model and those obtained after redefinition having learned the error with $M5'$, column $\%G$ represents the gain percentage.

δ	#=15			#I = 20			#I = 25		
	PC	$M5'$	%G	PC	$M5'$	%G	PC	$M5'$	%G
10	0.381	0.308	19.093	0.317	0.239	24.706	0.393	0.285	27.412
20	0.488	0.357	26.797	0.435	0.322	26.074	0.453	0.335	26.022
30	0.551	0.418	24.186	0.491	0.375	23.742	0.488	0.360	26.135
40	0.597	0.477	20.061	0.557	0.445	20.137	0.541	0.422	21.988
50	0.631	0.527	16.430	0.595	0.495	16.719	0.591	0.492	16.680
60	0.638	0.553	13.397	0.626	0.543	13.274	0.608	0.528	13.217
70	0.640	0.565	11.764	0.623	0.553	11.172	0.617	0.550	10.782
80	0.643	0.578	10.060	0.629	0.569	9.457	0.622	0.564	9.365
90	0.647	0.583	9.933	0.639	0.581	9.040	0.639	0.585	8.509

absolute error greater than or equal to 2 is 0.5%. It would therefore be interesting to use a different learning approach which is able to work with unbalanced data sets and use the RMSE metric rather than MAE as the objective function for the algorithms.

5.4. Q3: Identifying informative attributes

Finally, we have also undertaken an analysis to determine the contribution of individual features to the explanation. We would like to mention that this is not a problem of feature selection, i.e. finding a good subset of relevant features in order to optimize an accuracy metric (the objective function) since $M5'$ has its own built-in feature selection facility. Our objective, therefore, will be to investigate the impact of each selected feature on the 2D explanation interface. In this respect, we shall compare the performance of our model when using all the original attributes from the 2D explanation with the results obtained when the feature under study is excluded (omitting the feature) and the learning process is rerun. The marginal contribution of each attribute, at_i, can then be measured by considering the performance metrics for both situations, learning with all the features and learning using all but the selected attribute.

In order to conduct the study, we have set the parameters $\#I = 20$ and $\delta = 50$ for the $M5'$ algorithm when predicting a rating using the ELCF model. We have run a set of 33 evaluations, 29 of which were obtained when training the $M5'$

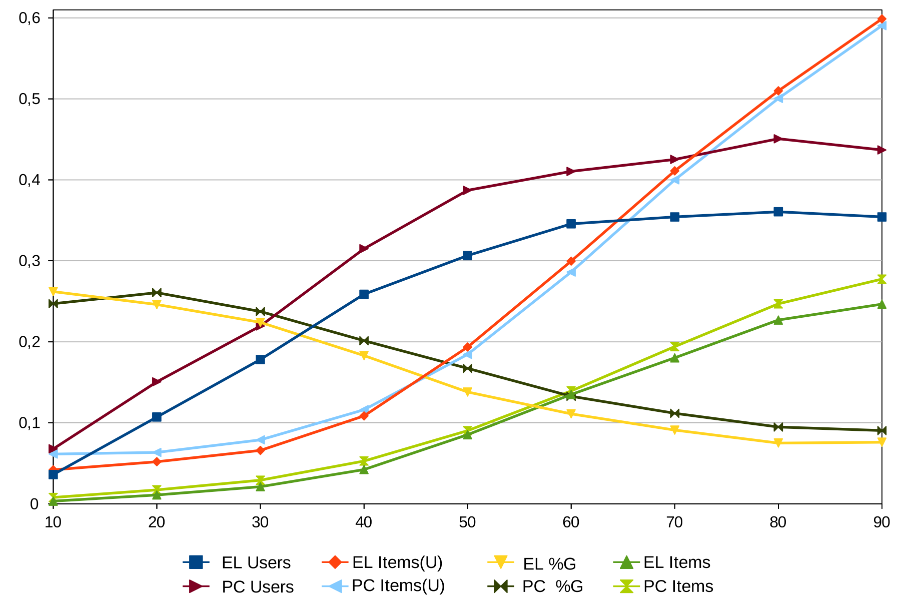

Fig. 8. Relative values for the ELCF (EL) and PCCF (PC) models. The *X*-axis present the δ threshold used and the *Y*-axis presents the relative gains when considering the number of users (Users), the number of triggered rules (when considering the items rated by the selected users (Items(U)) and also with respect to the total number of items (Items)) and the MAE improvements (%G).

Table 6
RMSE values using the original predictions for the ELCF (EL) model and those obtained after redefinition having learned the error with *M5'*.

δ	#I = 15			#I = 20			#I = 25		
	EL	*M5'*	%G	EL	*M5'*	%G	EL	*M5'*	%G
10	0.809	0.73	9.966	0.736	0.687	6.66	0.803	0.76	4.692
20	0.806	0.74	8.651	0.798	0.732	8.27	0.824	0.74	9.737
30	0.832	0.75	10.22	0.828	0.750	9.42	0.827	0.75	9.331
40	0.892	0.81	9.203	0.889	0.811	8.77	0.875	0.79	9.963
50	0.89	0.83	7.289	0.892	0.831	6.84	0.881	0.82	7.207
60	0.905	0.85	6.116	0.898	0.846	5.79	0.893	0.84	5.578
70	0.916	0.87	5.206	0.914	0.870	4.81	0.895	0.85	4.519
80	0.922	0.87	5.09	0.918	0.882	3.92	0.908	0.87	3.796
90	0.927	0.88	4.971	0.919	0.878	4.46	0.905	0.87	3.957

Table 7
RMSE values using the original predictions for the PCCF (PC) model and those obtained after redefinition having learned the error with *M5'*.

δ	#I = 15			#I = 20			#I = 25		
	PC	*M5'*	%G	PC	*M5'*	%GB	PC	*M5'*	%G
10	0.736	0.687	6.66	0.539	0.503	6.74	0.603	0.534	11.5
20	0.798	0.732	8.27	0.671	0.593	11.6	0.698	0.614	12
30	0.828	0.750	9.42	0.706	0.631	10.7	0.710	0.619	12.8
40	0.889	0.811	8.77	0.753	0.677	10.1	0.739	0.655	11.4
50	0.892	0.831	6.84	0.779	0.713	8.46	0.776	0.714	7.99
60	0.898	0.846	5.79	0.806	0.750	6.95	0.787	0.735	6.56
70	0.914	0.870	4.81	0.801	0.752	6.15	0.795	0.750	5.63
80	0.918	0.882	3.92	0.806	0.763	5.34	0.797	0.757	5.06
90	0.919	0.878	4.46	0.817	0.771	5.61	0.819	0.779	4.93

algorithm after excluding one of the features presented in Section 4.1 each time. Each of these data sets will be named with the label of the removed feature, for instance *lt_#s* represents the data set obtained after excluding the number of items in the left-top quadrant of the 2D interface. In addition, and for comparison purposes, we have run experiments with four additional subsets of features:

- "r, e, H": a new configuration with the rating $\hat{r}(a,o)$, the error in the prediction, $e(o)$, and the entropy, $H(o)$, as the only features in the training sets. It should be noted that this subset of features should represent the situation where no information was obtained from the 2D-based explanation.
- *_MAE*: obtained after removing all features related to MAE.
- $\_\eta$: obtained after excluding $\eta(o)$-related attributes.
- *_#S*: obtained after removing those features related to the number of items presented in the interface.

In order to comment on the results obtained and not overload the reader with large tables, we present the graph in Fig. 9 that will enable certain conclusions to be drawn. In this graph, we represent:

- Y-axis: the ratio of rules triggered when an attribute at_i is excluded in relation to the number of rules triggered considering all the attributes, i.e.

$$RR_{at_i} = \frac{\#rules(BSL \setminus at_i)}{\#rules(BSL)},$$

with *BSL* representing the model that considers all the features in the training sets. From a general perspective, we could say that after removing a feature from the original set, the lower the number of triggered rules, the more important this attribute is. Lower values of RR_{at} will therefore imply that the particular attribute has greater importance under this objective.
- X-axis: we also compute the ratio of the gain improvements obtained after removing an attribute, i.e.

$$GR_{at_i} = \frac{\%G(BSL \setminus at_i)}{\%G(BSL)},$$

where $\%G$ represents the gain obtained when comparing the error obtained with the predictions given by the original RS with those obtained after learning from explanation (RS + M5′). As before, the lower the gain ratio, the greater the importance of the removed attributes.
We would like to mention that since the exclusion of a feature affects the decision about whether to trigger a rule or not, and in order to allow the different accuracy results to be comparable, we have considered in the performance metrics the error in the predictions for all the items rated by the users, regardless of whether a rule had been triggered for this item or not.

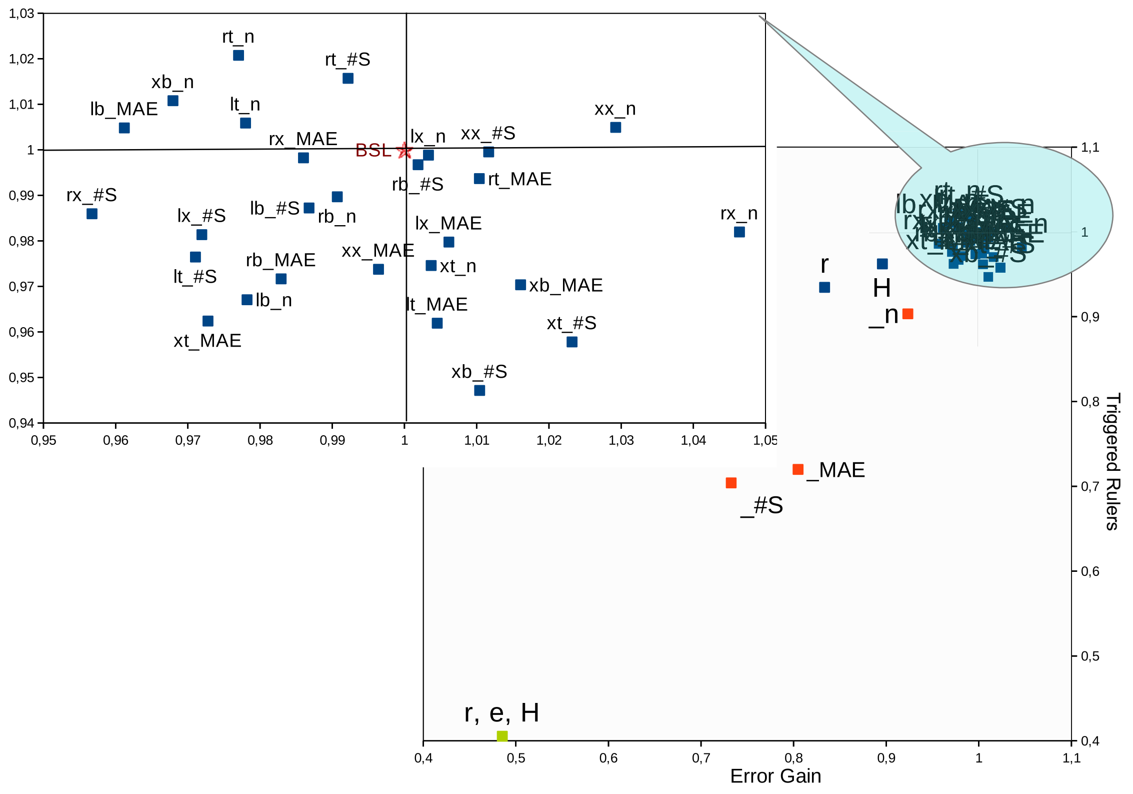

Fig. 9. Number of triggered rules for each considered set of features.

By observing the graph, we could then corroborate the importance of the 2D graph for explanation. By considering only "r, e, H" (attributes which are not based on the 2D graph) as selected features we execute 40% of the rules, losing around 50% of the gain. We are therefore able to say that there is valuable information for users in the 2D graph, enabling them to draw conclusions about the predicted rating. Regarding the grouped set of variables, we can say that both the location of the items on the 2D graph (represented by $\_\#S$) and the error in the predictions (represented by $\_\#MAE$) are equally important for learning purposes.

Focusing on individual features (represented in blue in Fig. 9), we can see that the rating (r) and the entropy (H) are the most important. This might explain why the user considers the histogram-based explanations to be highly valuable (they can infer information from these in order to understand the quality of a prediction). As we have seen, however, the information provided for these features can be highlighted with the knowledge presented on our 2D graph interface.

In terms of the features extracted from the 2D graph, we should mention that it is difficult to test whether the difference between two features regarding a measure is significant or not, mainly because certain features are highly related (for instance, the data used to compute $lb\_\#S$ are also used to compute $lx\_\#S$). Nevertheless, various general conclusions can be drawn: (i) global information (xx_*) seems to be less informative than local information, which considers the particular location of the items on the graph; (ii) the items predicted with low error (xt_MAE) are important for both performance metrics; (iii) the number of similarly rated items (those on the left-hand side, $lx\_\#S$, $lt\_\#S$ and $lb\_\#S$) are important; and (iv) there is, however, valuable information when the system does not predict properly (lower part of the graph, rb_MAE and rb_η) and in those situations where the items were rated differently to the target one (on the right-hand side) (rb_MAE, rb_η and $rx\_\#S$). This could confirm our intuition that in order to reason about the quality of a recommendation, we must consider both right and wrong predictions. If we are interested in redefining erroneous predictions, it therefore becomes necessary to analyze system performance in those situations where it is not working properly.

Finally, we wish to present the five most relevant attributes that appear in the triggered rules when all the features in the training sets are considered (we show in brackets the percentage of times each attribute appears in the triggered rules, setting the δ threshold to 50 and #I = 20): the predicted rating, r, [50%]; entropy H [25%]; xb_MAE [18%]; lt_MAE [17%] and xt_MAE [16%]. These data support our above conclusions, i.e. the most relevant attributes are those relating to the target item (the predicted rating and the entropy) and the quality of the predictions over previous ratings.

6. Concluding remarks and future lines of work

In this paper, we have shown that explanations can be considered as a valuable source of knowledge that can be exploited by an RS. This paper offers a first insight into this topic and, more specifically, our research focuses on using neighbors' opinions about items previously rated by the user in the explanations. We demonstrate that relevant data can be gathered from this type of explanation, that we can use machine-learning strategies to change this data into knowledge (in terms of rules) that can prove useful for improving the RS. Using this approach, we can explore the benefits that can be obtained after analyzing a given explanation, something which has already been identified in literature [21] but has never been properly addressed.

Our experimentation shows that the advantages in terms of improved accuracy of using explanations is highly user-dependent: what is good for one user is not necessarily good for another. We have identified a series of features that can be used for prediction purposes and which can be used to improve recommendations for around 30% of users by considering previous user experiences.

During the course of this paper, we have identified various problems which are worthy of further research. Among these, we wish to highlight the use of this approach when considering different recommendation algorithms. We believe that this is particularly useful for a context-based recommendation. In this case, the features should take into account statistics about the presence of tags in the item's description.

We also consider it relevant to investigate in greater depth the reasons why certain users might find a given explanation useful or not. This might help gain a better understanding of user preferences or tastes, something which is highly relevant for the RS community.

Finally, we plan to focus on two particular problems which are directly connected with this paper. Firstly, we plan to research the use of different machine-learning strategies. Our idea is to focus on the detection of large errors in recommendations to enable algorithms to work with unbalanced data sets. In this respect, we propose the use of the RMSE metric as an objective function. Secondly, we believe that the results obtained from this kind of research can be used to develop better explanation interfaces.

Acknowledgments

This work was jointly supported by the Spanish Ministerio de Educación y Ciencia and Junta de Andalucía, under projects TIN2011-28538-C02-02 and Excellence Project TIC-04526, respectively, and also the AECID fellowship program.

References

[1] M. Bilgic, R.J. Mooney, Explaining recommendations: satisfaction vs. promotion, in: Proceedings of Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research at the 2005 International Conference on Intelligent User Interfaces, San Diego, CA, 2005.
[2] W. Chen, W. Hsu, M.L. Lee,. Tagcloud-based explanation with feedback for recommender systems, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2013, pp. 945–948.
[3] S. Cleger-Tamayo, J.M. Fernández-Luna, J.F. Huete, A new criteria for selecting neighborhood in memory-based recommender systems, in: Proceedings of the 14th International Conference on Advances in Artificial Intelligence: Spanish Association for Artificial Intelligence, CAEPIA'11, Springer-Verlag, 2011, pp. 423–432.
[4] S. Cleger-Tamayo, J.M. Fernandez-Luna, J.F. Huete, Explaining neighborhood-based recommendations, in: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, ACM, New York, NY, USA, 2012, pp. 1063–1064.
[5] S. Cleger-Tamayo, J.M. Fernández-Luna, J.F. Huete, N. Tintarev, Being confident about the quality of the predictions in recommender systems, in: Proceedings of the 35th European Conference on Advances in Information Retrieval, ECIR'13, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 411–422.
[6] D. Cosley, S.K. Lam, I. Albert, J.A. Konstan, J. Riedl, Is seeing believing? How recommender system interfaces affect users' opinions, in: CHI '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 2003, pp. 585–592.
[7] H. Cramer, V. Evers, S. Ramlal, M. Someren, L. Rutledge, N. Stash, L. Aroyo, B. Wielinga, The effects of transparency on trust in and acceptance of a content-based art recommender, User Model. User-Adapt. Interact. 18 (5) (2008) 455–496.
[8] G. Friedrich, M. Zanker, A taxonomy for generating explanations in recommender systems, AI Mag. 32 (3) (2011) 90–98.
[9] F. Gedikli, D. Jannach, M. Ge, How should I explain? A comparison of different explanation types for recommender systems, Int. J. Human Comput. Stud. 72 (4) (2014) 367–382.
[10] D. Goldberg, D.A. Nichols, B.M. Oki, D.B. Terry, Using collaborative filtering to weave an information tapestry, Commun. ACM 35 (12) (1992) 61–70.
[11] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, J. Mach. Learn. Res. 10 (2009) 2935–2962.
[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The Weka data mining software: an update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[13] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, third revised ed., Morgan Kaufmann, 2011.
[14] J.L. Herlocker, Position statement – explanations in recommender systems, in: CHI'99 Workshop, Interacting with Recommender Systems, 1999.
[15] J.L. Herlocker, J.A. Konstan, J. Riedl, Explaining collaborative filtering recommendations, in: CSCW '00: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, ACM, New York, NY, USA, 2000, pp. 241–250.
[16] J.L. Herlocker, J.A. Konstan, J. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, Inf. Retrieval (5) (2005) 287–310.
[17] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. (22) (2004) 5–53.
[18] A. Hernando, J. Bobadilla, F. Ortega, A. Gutirrez, Trees for explaining recommendations made through collaborative filtering, Inf. Sci. (239) (2013) 1–17.
[19] J. Huete, J. Fernández-Luna, L. de Campos, M. Rueda-Morales, Using past-prediction accuracy in recommender systems, Inf. Sci. (199) (2012) 78–92.
[20] B. Knijnenburg, M. Willemsen, Z. Gantner, H. Soncu, C. Newell, Explaining the user experience of recommender systems, User Model. User-Adapt. Interact. 22 (4–5) (2012) 441–504.
[21] J.A. Konstan, J. Riedl, Recommender systems: from algorithms to user experience, User Model. User-Adapt. Interact. 22 (1–2) (2012) 101–123.
[22] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[23] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, W.-K. Wong, Too much, too little, or just right? Ways explanations impact end users' mental models, in: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2013, pp. 3–10.
[24] N. Lathia, S. Hailes, L. Capra, X. Amatriain, Temporal diversity in recommender systems, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, ACM, New York, NY, USA, 2010, pp. 210–217.
[25] S.M. McNee, J. Riedl, J.A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: CHI '06 Extended Abstracts on Human Factors in Computing Systems, CHI EA '06, ACM, New York, NY, USA, 2006, pp. 1097–1101.
[26] D. McSherry, Explanation in recommender systems, Artif. Intell. Rev. 24 (2) (2005) 179–197.
[27] MovieLens dataset. <http://www.grouplens.org/data/>.
[28] F. Ortega, J. Bobadilla, A. Hernando, F. Rodríguez, Using hierarchical graph maps to explain collaborative filtering recommendations, Int. J. Intell. Syst. 29 (5) (2014) 462–477.
[29] A. Papadimitriou, P. Symeonidis, Y. Manolopoulos, A generalized taxonomy of explanations styles for traditional and social recommender systems, Data Min. Knowl. Disc. (2011) 1–29.
[30] P. Pu, L. Chen, Trust-inspiring explanation interfaces for recommender systems, Knowl.-Based Syst. 20 (6) (2007) 542–556.
[31] J. Quinlan, Learning with continuous classes, in: 5th Australian Joint Conference on Artificial Intelligence, World Scientific, Singapore, 1992, pp. 343–348.
[32] P. Resnick, H.R. Varian, Recommender systems, Commun. ACM 40 (1997) 56–58.
[33] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, US, Boston, MA, 2011.
[34] M. Rossetti, F. Stella, M. Zanker, Towards explaining latent factors with topic models in collaborative recommender systems, in: Proc. of the International Workshop on Database and Expert Systems Applications, DEXA, 2013, pp. 162–167.
[35] A. Sharma, D. Cosley, Do social explanations work? Studying and modeling the effects of social explanations in recommender systems, in: Proc. of the International World Wide Web Conference, WWW, 2013, pp. 1133–1143.
[36] Y. Shi, M. Larson, A. Hanjalic, Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges, ACM Comput. Surv. 47 (3) (2014) 1–45 (No. 1. Article 3).
[37] J. Sormo, F. Cassens, A. Aamodt, Explanation in case-based reasoning perspectives and goals, Artif. Intell. Rev. 24 (2) (2005) 109–143.
[38] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, Providing justifications in recommender systems, IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans 38 (6) (2008) 1262–1272.
[39] P. Symeonidis, A. Nanopoulos, A. Krinis, GeoSocialRec: explaining recommendations in location-based social networks, Adv. Databases Inf. Syst. Lect. Notes Comput. Sci. 8133 (2013) 84–97.
[40] A. Tejeda-Lorente, C. Porcel, E. Peis, R. Sanz, E. Herrera-Viedma, A quality based recommender system to disseminate information in a university digital library, Inf. Sci. (261) (2014) 52–69.
[41] N. Tintarev, J. Masthoff, Designing and evaluating explanations for recommender systems, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2011, pp. 479–510.
[42] N. Tintarev, J. Masthoff, Evaluating the effectiveness of explanations for recommender systems, User Model. User-Adapt. Interact. 22 (4–5) (2012) 399–439.
[43] Y. Wang, I.H. Witten, Induction of model trees for predicting continuous classes, in: Poster Papers of the 9th European Conference on Machine Learning, Springer, 1997.
[44] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, third ed., The Morgan Kaufmann Series in Data Management Systems, 2011.