Applied Intelligence 39 (2013) 3, pp.545-563

DOI: 10.1007/s10489-013-0432-x

Statistical user model supported by R-Tree structure

Javier Calle · Leonardo Castaño · Elena Castro · Dolores Cuadra

Abstract This paper is about developing a group user model able to predict unknown features (attributes, preferences, or behaviors) of any interlocutor. Specifically, for systems where there are features that cannot be modeled by a domain expert within the human computer interaction. In such cases, statistical models are applied instead of stereotype user models. The time consumption of these models is high, and when a requisite of bounded response time is added most common solution involves summarizing knowledge. Summarization involves deleting knowledge from the knowledge base and probably losing accuracy in the medium-term. This proposal provides all the advantages of statistical user models and avoids knowledge loss by using an R-Tree structure and various search spaces (universes of users) of diverse granularity for solving inferences with enhanced success rates. Along with the formalization and evaluation of the approach, main advantages will be discussed, and a perspective for its future evolution is provided. In addition, this paper provides a framework to evaluate statistical user models and to enable performance comparison among different statistical user models.

Keywords User model · User group model · Collaborative user model · Statistical user model · R-Tree · Benchmark

J. Calle · L. Castaño (✉) · E. Castro · D. Cuadra
Computer Science Department, Carlos III University of Madrid, Madrid, Spain
e-mail: lcastano@inf.uc3m.es

J. Calle
e-mail: jcalle@inf.uc3m.es

E. Castro
e-mail: ecastro@inf.uc3m.es

D. Cuadra
e-mail: dcuadra@inf.uc3m.es

1 Introduction

One of the specific applications of user modeling is that regarding human-like interaction systems. These systems seek to imitate human behavior through interaction, but the final purpose of human like interaction is also to provide users experiences and responses fitting their specific background and objectives. That is, provide contents and responses adapted to every single user need. The most common approaches to provide such features and contents are statistical and predictive user models. In general, user modeling provides adaptation to the interlocutor, predicting his features and preferences and advancing his probable behavior. Several features within human-like interaction cannot be modeled by a domain specialist as it is only possible to model these features by observing thousands of real cases or by a trial and error strategy. An example of these types of features is the selection of dialogue strategies during the interaction, or the emotional behaviors. That is the interaction depends on the interlocutor, some interlocutors might prefer longer interactions while other interlocutors might prefer short questions. Therefore too many cases might occur and only two different approaches can deal with this scenario, statistical user model or trial and error strategies. Even though, statistical user models can deal with this situation, they can also be applied to domains that can be modeled by a domain specialist.

User models usually rely on hand crafted knowledge basis that can be difficult to construct and almost non-adaptable or non-extendable. However, statistical user models build their knowledge base according to the experience granted in previous sessions. In addition, most commonly in predictive and statistical user models, the dependent parameter signifies a user preference or behavior.

The present proposal involves the development of a specific group's user model that is able to predict unknown

user information, dealing with the absence of comprehensive information. This user model comes to fill an empty space in current state of the art, focusing not only user models based on stereotypes, but also dealing with the problem of response time constraints and knowledge loss which is a common disadvantage within group user models. Knowledge Bases fed by experience through the system's lifetime grow to become unmanageable in most cases, bringing down the system's performance. To preserve efficiency over time, many systems apply knowledge summarization at the expense of knowledge quality and accuracy in the inferences drawn from it. This paper proposes a solution to this problem based on an advanced storage structure (R-tree [27]).

After the proper review of related works, a proposal for statistical user modeling without knowledge loss is described, along with another somewhat analogous model that has knowledge loss. Then, their performance will be compared through the evaluation section, to finally obtain some conclusions and the perspective of the potential applications of the proposal. To achieve this evaluation a user is considered as a set of features, at every moment, a sub-set of features of the interlocutor are known and the system tries to predict the value of an unknown feature. In addition a review of main frameworks to compare user models performance has been achieved, identifying a lack of benchmark to evaluate statistical user models. Regarding this lack of benchmark to evaluate and compare statistical user models, this paper proposes a complete framework to evaluate statistical user models and achieve performance comparisons among different user models for user modeling. The need of a benchmark to evaluate statistical user models resides on the complexity of this task and the need of making evaluation results comparable within the community of statistical user models for user modeling.

2 State of the art in user modeling

User Modeling seeks to characterize the user for adapting the interaction to his needs and preferences. Traditional research on adaptation and User Modeling is frequently supported by the knowledge provided by some expert in the application domain. This approach has produced remarkable achievements in diverse fields, such as user classification for Dialogue Management [38], user modeling via stereotypes [41], and group user models [24]. However, traditional user modeling requires hand-built knowledge bases with enough representative points, which is a hard and costly task, apart from other drawbacks as the risks of vitiated data and certainty degradation. Besides, in some domains is difficult to have an expert (maybe even impossible). Alternatively, the statistical approaches emerge [55] relying on the experience of the system (past interactions with other users), thereby avoiding the need of an expert.

According to Larson [33] statistical models are intended to perform predictions of a dependent parameter, departing from observed samples. Regarding user modeling, such parameter represents an aspect of a user future behavior, likes or preferences. The usage of statistical and probabilistic user models has been manifested by the increasing publication of research in these lines [8, 50]. However, for several years it is claimed that those approaches are still far of being applicable and effective in some fields [51], and therefore there is still room for research in this area. Albrecht and Zukerman [3] identified several challenges that user modeling presents to statistical and probabilistic modeling techniques, classifying them into three categories: limitations of current user modeling approaches, dynamic nature of user modeling data and efficiency considerations.

The statistical focus has been largely applied to recommender systems, seeking the item (product, information, content ...) that suits best the user characterization by comparing it with past users. Given a specific set of items, the recommender system has to rate their usefulness/likeness for the user (who has not experienced them yet) by means of a utility function [1]. Since that function is not defined for all products on the whole universe of potential users, the recommenders need to extrapolate for accomplishing that rating and then select the most useful item (or N most useful items). Estimations usually apply heuristics or follow some criterion such as minimizing the mean square error on large knowledge bases acquired from experience.

Some recommender systems also take into account contextual information affecting the interaction, such as the type of user device (laptop, mobile, etc.). Therefore, the recommended content may differ depending on the capabilities of the device or on its screen size [10]. The multi-agent approach is presented as suitable for this requirement, enabling a client-agent for each different device, as in the systems Masha [44] and Muaddib [43]. Multi agent technology has repeatedly supported recommender systems [15], bringing some of the problems of this type of environments. Communication between agents, apart from being a time-consuming task, is a major obstacle when they are supported by different ontology. Endowing the system with inter-ontology capabilities [42] or semantic negotiation [22] are some solutions to this problem. Besides, Ontology support is a useful way of enhancing the model capabilities [23], unifying different terms for the same concept and enriching the available information regarding the current user.

Next, the recommender systems where the statistical user model is embedded have been classified according to their approach to estimate ratings. In first place, the content-based approach [6] predicts user behavior through the past behavior based on Information Retrieval methods such as survey techniques or implicit methods to obtain and store profiles about user preferences and likes [5]. Some problems

have been detected: the filtering and partial content analysis [46], the over-specialization [47, 54], and the new user problem [52]. This latter issue, doing predictions on new users or known users in new situations, can be alleviated applying some profile expansion techniques based on query expansion methods from the Information Retrieval area [25]. As an answer to this problem, also appeared the collaborative filtering [40].

In the collaborative approach, predictions on current user are built from the behavior of other like-minded people. Thus, collaborative filtering is used under the assumption that a user behaves similarly to other user. Such techniques have been successfully applied for recommending books [34], news [14], or just to provide customers with personalized suggestions [37]. There can be identified two subtypes of collaborative techniques: the memory-based, using the whole or a subset of the user-feature database (user-feature matrix) to compute similarities among users or items [45]; and the model-based techniques, relying on patterns extracted from a training set to extrapolate reliable predictions for the real world [16]. Current challenges in collaborative filtering include the data sparsity problem [11], the scalability in collaborative filtering [34] and the gray sheep problem [26]. The sparsity data problem happens in sparse databases when two similar users are identified as very dissimilar users just because they both have not rated the same items [11], and can be alleviated using dimensionality reduction techniques [12]. The scalability regards huge knowledge bases acquired from experience, which become unmanageable and lead to high response times. As a solution, knowledge can be summarized, but such procedure involves a certain knowledge loss. Finally, dealing with unclassifiable users (gray sheep) can be solved by weighting k-nearest neighbors (yet distant neighbors), often at the expense of certainty in the predictions.

Finally, the hybrid recommender systems combine collaborative learning techniques with other recommendation methods to mitigate some problems of individual recommendation techniques and improve performance. Most frequent hybridization is that obtained by merging the collaborative and content-based approaches, but there exist other systems, such as those in [32], that combine collaborative filtering with demographic recommenders based on profiles. Several classifications of hybrid recommender systems can be found, being the most extended criterion the hybridization method [13, 17] which classifies them in weighted, switching, mixed, feature combination, cascade, feature augmentation and meta-level.

The pair content-based/collaborative always suffers the *start-up problem* (lack of initial knowledge) since both approaches need certain volume in their knowledge bases to provide reliable predictions. However, hybridization may enable inference from sparse databases and somehow complete their content. Several researchers have compared the performance of hybrid recommenders with pure collaborative filtering and found that hybrid proposals can make better predictions especially regarding the new user problem and when dealing with sparse databases [9]. On the other hand, main drawback of hybrid systems is the complexity of their implementation [39].

When evaluating any recommender system, two main issues have to be addressed: the data sets and the selection of proper metrics. Regarding the first, the nature of the dataset may determine the performance of the model. In fact, several collaborative systems are designed for a given dataset structure and even a specific dataset [35], which deviates from the genericity pursued in this type of model. The selection of a metric may also depend on the domain of evaluation [4].

On the subject of methods and metrics for this sort of evaluation, a widespread practice is found in comparing systems with older versions of the same kind. However, some authors state that results are not significantly different when both systems are tuned to its maximum. Furthermore, [31] also suggest that most users provide inconsistent ratings when asked to rate the same item at different times. The main difficulty in selecting one or more metrics to evaluate collaborative filtering algorithms is the lack of standardization. New works in the area are still introducing new methods and metrics, hindering the comparison with similar works and the detection of differences between the inspected systems [30]. That work classifies the subjected metrics into predictive accuracy metrics, classification accuracy metrics, and rank accuracy metrics. Predictive accuracy metrics are based on comparisons between *true user ratings* and system's predictions to measure the distance between them [16]. Secondly, classification metrics are suitable for recommenders aimed to classify items as *good or bad* for the user. Both metrics measure the ability of any recommender to make correct or incorrect decisions [9, 29]. Finally, rank accuracy metrics are suitable for systems providing an ordered list of relevant items to the user [16, 28, 53].

Finally, there are some other considerations relevant for the evaluation process. Cosley et al. [21] found that users are sensitive to prediction's accuracy (the predicted rating can affect the current rating given by the user). Besides, information on why some item has been recommended might help users to understand the reason of the suggestion and thus increase their confidence in the recommender system.

Summarizing, content-based and collaborative filtering systems must deal with several problems like the new user problem, the over-specialization problem, the data sparsity problem, the gray sheep problem, and the scalability and limited response time problems. Some of them regard main challenges in the area as identified by Albretch and Zukerman, as the efficiency considerations and the dynamic user features, for which there is a long way to go.

3 Proposal

Departing from the previous analysis, the approach that is going to be described during this section is a statistical and predictive user model of groups and totally domain-independent. Therefore, our approach builds a knowledge base of groups based on past sessions held with different users.

During an interaction with a user, the system is able to acquire information atoms about user features (explicitly stated by the interlocutor and biometrics acquired by interfaces), and user behavior and preferences. At the end of the session, the collection of information atoms is stored in the knowledge base as an individual user, (although the same user can hold several sessions that will be stored independently). Additionally, when the system takes advantage of any unknown information atom (for example, a shopping preference), the model can predict it by comparing the available information about the current user with the users descriptions stored in the knowledge base and then inferring that unknown atom. One of the most important problems of this approach is the excessive growth of the knowledge base, which leads to significant performance loss. It should be reminded that the response time in interactive systems is a critical factor, and the response should not be delayed. In order to reduce the knowledge base down to a few individual descriptions, the search for a solution leads to consider user groups, represented by group descriptions, not user descriptions, bringing together several session descriptions which are found to be very similar (or at least the most similar ones among those stored in the base). The problem that arises at this point is the loss of knowledge produced by summarizing processes. The subsequent user sessions will be harder to match to an individual within the base, thus predictions will lose both accuracy and certainty.

This paper introduces a statistical user model able to avoid such loss of knowledge by observing several hierarchical sub-universes of users. The proposal is later evaluated by comparing its performance with a simpler model that runs those fusion procedures (merging similar groups of users) for keeping the knowledge base size within a reasonable range (hence assuming that loss of knowledge).

To improve the understanding of the proposal and its scope, this section will define the two models (with and without loss of knowledge, respectively). In the common functions (individual matching, fusion, etc.) a similar formulation will be applied, as the core of the proposal is to present the focus and the structure of this new model headed to avoid loss of knowledge, and to analyze the advantages in accuracy and certainty that this approach provides.

3.1 User model subject to knowledge loss

In this case, the knowledge growth is to be controlled by setting bounds to the amount of knowledge. This approach of a user model subject to knowledge loss (KLUM) is also described in [18]. Specifically, this model will count on an upper bound for the number of descriptions of user groups, and another one for the total number of information atoms. The first one reduces the number of candidates when matching groups of users, while the other shortens the average execution time of each matching. Their foundations and use will be further explained later in this section. The bounds are empirically obtained through preliminary experimentation by observing the response time requirements for the interaction system and the performance of the model (with the real hardware resources) with different bound configurations.

This model relies upon a few notions (some of them very common) which are now to be defined. Let's have a user description (*UD*) (1) defined extensively as a set of information atoms represented by tuples of the type {feature, value, sign, certainty}.

Let *UD* be $\subset \{(c, v, s, z)$: c is a user feature,

$$v \in domain(c), s \in \{-1, 1\}, z \in (0, 1]\} \quad (1)$$

Each tuple in a given *UD* has a direct meaning characterizing that user: the value applies (or not, depending upon the sign) to the user regarding the feature, observing that this piece of information has a given certainty, that is, a measure of the confidence of being certain when applying this statement. The term certainty is preferred to probability since there are diverse sources of knowledge including the user's fuzzy assertions through his/her interventions. The certainty must be higher than zero, given that value zero is considered as 'no information', and any tuple with such certainty will be omitted. Any *UD* may lack tuples for a given feature (which is equivalent to including a tuple with certainty zero for each value and sign in its domain) or include several rows with the same feature but different values.

The *UD* includes no identifier, so this description initially defined to stand for a user may actually represent several similar users. Consequently, a stereotype (henceforth, group of users) is also a *UD* with the addition of a value for its 'population', indicating the number of sessions held to acquire this group. Notice that the term *session* is here preferred to the term *user*, since there are no identifiers and therefore all those similar *UD* (or part of them) could have been acquired through several interactions (sessions) held with the same user. Formally, the definition of a group of users is (2):
Let a group of users (stereotype) be

$$GU \equiv (UD, p) \mid p \in N \quad (2)$$

The whole set of groups of users for a particular domain will be named universe of users (*UU*). The *UU* defined here is in fact the currently available knowledge about the universe of users, stored in the knowledge base of the

Table 1 Partial match function (6)

μ-match type	Condition	Formula
Disjoint	given $x = (c, v, s, z)$ and $GY = (Y, pY)$ $\forall y = (c', v', s', z') \in Y$, meets $c \neq c'$	$\mu(x, GY) = 0$
Coincident	given $x = (c, v, s, z)$ and $GY = (Y, pY)$ $\exists y = (c, v', s', z') \in Y$	$\mu(x, GY) = \overline{\delta(x, y)}$, $\forall y \in Y$

model's implementation. Since it is a statistical model, any new session will provide new knowledge. Consequently, the description of the *UU* within any given implementation is dynamic, and the *GU*s can be updated or new ones can be added. However, an extreme growth of the knowledge base could cause a fatal performance drop. To prevent such drawbacks, this model includes upper bounds to the number of information atoms (η) and to the number of groups of users (γ) within a *UU*. The following expression (3) defines the *UU*, taking into account the definition of group of users and the definition of *UD*:

$$\text{Let a universe of users be } UU \equiv \{GU_i\} \mid i \in N[1, \gamma] \tag{3}$$

During any interaction, the interlocutor is characterized incrementally as information atoms about him/her are acquired. The atoms are gathered in the current user description state (*CUDS*), which is formalized as a general *UD* (1) according to the expression (4).

$$\text{Let the } CUDS \text{ be a } UD, \text{ and its resultant stereotype}$$
$$GU(CUDS) = (CUDS, 1) \tag{4}$$

When finishing a session, the *CUDS* is stored as a new group of users (with population 1). Storing every new *CUDS* in the knowledge base may lead to exceed the threshold (γ). In such cases, the knowledge has to be summarized to meet γ again. Specifically, the most similar pair of *GU*s has to be merged into a single *GU*, thus decreasing the size of the *UU* in a unit. The required procedures to do this are 'finding the proper pair' and 'merging groups', which are based on the match and the fusion functions, respectively. These functions are characteristic of the model.

For searching the most suitable pair of *GU*s to be merged, all candidate pairs (each pair of *GU*s in the knowledge base) will be evaluated with the match function (M) which provides a value in the range [0, 1] measuring the similarity between those groups. The pair maximizing that value will be chosen for merging. The M function proposed here is described in (5).

$$\begin{aligned} &\text{Let the match be } M : UU \times UU \to [0, 1] \mid M(G_X, G_Y) \\ &= \left(\left\{\overline{\mu(x, G_Y)}, \overline{\mu(y, G_X)}\right\} + 1\right)/2 \\ &\quad \forall x \in X \; \forall y \in Y, \text{ where } G_X = (X, p_x) \\ &\quad \text{and } G_Y = (Y, p_y) \text{ are groups of users} \end{aligned} \tag{5}$$

The M function is defined upon the notion of partial matching (μ), providing a portion of information about the matching. It is used for comparing an information atom (from a *GU*) with another *GU*. The result can be positive or negative, that is information about matching in case of positive partial matching or information about knock out in case of negative partial matching. The definition of this function observes two main cases: the disjoint partial matching (special case), and the coincident partial matching (general rule).

In first place, the disjoint partial matching occurs when comparing an atom on a feature that is missing in the other *GU* (there is no atom defined for that feature). These cases provide no information about the matching, and therefore the result of disjoint partial matching is zero. The rest of the cases are considered as coincident partial matchings, and its calculation has to observe every atom in the *GU* defined on the coincident feature. Thus, its definition will be based on another function, the single atom match (δ) providing a measure of the match for two individual atoms. The partial match function (6) is defined in Table 1 as observed in the aforementioned cases.

The definition of the proposed δ function (atoms match) is divided into three different components. The first component is the sign of the matching (7) that is a notion of lace or a knock out notion and is calculated as follows:

$$S = S_x \cdot S_y \tag{7}$$

being S_x, S_y the signs of the atoms to compare. Even though S defines the sign of the result; the sign of match through two cases should also be observed: when both atoms have the same value (v), the match is equivalent and the sign is positive; on the contrary, if the values are different the match is non-equivalent and the sign is negative. The second component of partial matching function involves weighting the certainties of both atoms. The measure used to assign weights to these certainties is the arithmetic average of both certainties. This component is calculated in the following expression (8), according to the next formula:

$$\overline{Z} = (z_x + z_y)/2 \tag{8}$$

The third component of partial matching is going to be applied when the equivalency is about the antithesis (both atom signs are negative) or the non-equivalency involves a

Table 2 Atom matching cases (10)

Atom match type	Sign (x)	Sign (y)	Formula
Equivalent: $x = (c, v, s, z)$ and $y = (c, v, s', z')$	+	+	$\delta(x, y) = S \cdot \overline{Z}$
	+	−	
	−	+	
	−	−	$\delta(x, y) = S \cdot \overline{Z} \cdot f$
Non-equivalent: $x = (c, v, s, z)$ and $y = (c, v', s', z')$ and $v \neq v'$	+	+	$\delta(x, y) = -S \cdot \overline{Z}$
	+	−	$\delta(x, y) = -S \cdot \overline{Z} \cdot f$
	−	+	
	−	−	

negative value (any sign is negative). In these cases, the result must be corrected by adding a domain-dependent correction factor (9). The information provided by the partial matching excludes one element of our universe; therefore the correction factor is calculated as follows:

$$f = 1/\left(\left|\mathrm{dom}(C_x)\right| - 1\right) \tag{9}$$

Once the three components have been explained, their product defines the magnitude of the atom matching formula (10). Taking also into account the cases of coincidence (equivalent and non-equivalent), the function can be defined as shown in Table 2.

As stated before, when the number of GUs within the UU exceeds the upper bound (γ) the most similar pair of groups are chosen by calculating the match of every candidate pair. Once chosen, the attention is focused on the fusion function (11), which can merge two or more user groups (in general, any non-empty part of a UU) into a unique user group. The resulting group will be added while the operated groups will be removed, thus decreasing the current number of groups in the knowledge base.

Let the fusion function be φ: $P(UU) - \phi \rightarrow GU \mid$

$$\varphi(G_1, \ldots, G_n) = GR = (R, p_R), \quad \text{where } G_i = (X_i, p_i),$$

$$p_R = \sum_{i=1}^{n} p_i \text{ and } R \text{ is}$$

$$\left\{ \left(c, v, \mathrm{sgn}(z), |z|\right) / \forall c, v \text{ meeting } (c, v, \_, \_) \in \bigcup_{i=1}^{n} X_i \right.$$

$$\left. \text{and } z = \left(\sum_j s_j \cdot z_j \cdot p_i \right) / p_R \; \forall i, j \text{ with } (c, v, s_j, z_j) \in X_i \right\} \tag{11}$$

It should be pointed out that certainty may drop severely when running reiterative fusions over several groups, producing unworthy tuples with virtually no effect on predictions. Besides, the excessive growth of tuples in the base may produce performance drop during inferences, and because of this an upper bound is set on the number of tuples (η). The purge procedure will simply remove as many tuples as needed to meet that bound again, starting always with the lower certainty tuples.

On the other hand, in order to provide prediction services, the user model is also able to infer values for unknown features of the user. It will do this by matching the *CUDS* (4) with each group stored in the knowledge base. Then a value is chosen by applying the selection function (ι). In the following chart ((12) and (13)), two selection functions ι_1 and ι_2 are proposed. Both of them will be observed in the evaluation section. The first approach is based on the idea that the most similar group will provide most reliable inferences. This approach is called maximum fit inference (12) because inferences are accomplished over the most similar group. On the other hand, the second approach is based on the idea that sometimes the most similar group does not provide the most reliable inference while a slightly different group might provide better inferences (13). This mechanism takes into account the match value and the certainty of the feature. Both proposals will be observed in the evaluation section.

$$\begin{aligned} &\iota_1(c, CUDS, UU) = v \mid \exists (c, v, s, z) \in X_i, GU = (X_i, p_i) \\ &\quad \in UU, \forall \left(c', v', s', z'\right) \in X_i \rightarrow z' \leq z, \\ &\forall GU' \in UU \rightarrow M\left(CUDS, GU'\right) \leq M(CUDS, GU) \end{aligned} \tag{12}$$

$$\begin{aligned} &\iota_2(c, CUDS, UU) \\ &\quad = v \mid \exists (c, v, s, z) \in X_i, GU = (X_i, p_i) \in UU, \\ &\forall \left(c', v', s', z'\right) \in X_i, \forall GU' \in UU \rightarrow z' \cdot M\left(CUDS, GU'\right) \\ &\quad \leq z \cdot M(CUDS, GU) \end{aligned} \tag{13}$$

Such a user model subject to knowledge loss is formalized in KLUM $(F, \gamma, \eta, M, \varphi, \iota)$, where F is a set of pairs (feature c, $domain(c)$) within a wide range of attributes and behaviors characterizing the user; γ and η are upper bounds to the number of GUs and to the number of tuples in the base, respectively; and M, φ and ι are the match, fusion and

inference functions, respectively. Notice that, although the set of features and domains F is included for definition completeness, the model can learn new features and domains as they are acquired through different sessions during the system's lifetime. Along with the model definition, some examples for those elements have been proposed, hence the depiction of a complete KLUM.

3.2 R-Tree based user model

The second approach involves developing a new proposal avoiding knowledge loss. This proposal is based on an R-Tree structure [27] and is called R-Tree User Model (RTUM). This data structure hierarchically divides the space into overlapped sub-spaces. Therefore the universe of users will be split into several sub-spaces (tree nodes) according to the user's similarities. A sub-universe of the user (14) is defined as follows:

Let a sub-universe of users j of the level i be

$$UU_{ij} \equiv \{GU_k\} \mid k \in N[1, n] \tag{14}$$

According to the R-Tree performance, new elements (individual descriptions of user sessions) will be added to the leaves of the tree. The rest of the nodes in the tree (non-leaf nodes) will keep the user's descriptions representing the set of users in its sub-tree ($UU_{i,j}$). Therefore, it can be stated that the universe of users in this approach is represented with different detail at each level of the tree, from the root node to the conjunction of all user's descriptions in the leaves of the tree. Therefore, the most detailed description is found in the conjunction of all users of level n where n is the level of the leaves in the tree (15). Formally:

$$\text{Universe of users } UU \equiv \bigcup_j UU_{n,j} \tag{15}$$

being n the tree's depth.

According to previous definition, it can be concluded that the rest of the information in the tree (non-leaf nodes) is redundant information. Thus, two types of elements can be identified, individual user's descriptions (at leaf nodes) and briefs of a sub-universe of users (at non-leaf nodes). The definitions of these elements ((16) and (17), respectively) are supported by a common user description (*UD*), which is the same already defined for the former model (1). The individual user group (16) found exclusively at the leaves has a *UD* and population 1, while the general user group (17) has population p and a pointer to a sub-tree representing the sub universe of users UU_{x+1}, and summarized within its *UD*.

Let GU be a group of users in the leaves level,

$$GU \equiv (UD, 1) \tag{16}$$

Let GU be a group of users in a non-leaf level,

$$GU \equiv (UD, p, UU_{x+1,y}),\ p \in N \tag{17}$$

Consequently, every node in the tree (except for the root) has a summary of its information in its ancestor node, and the summary has a pointer associated to that node. Formally:

$$\forall UU_{x,j} \text{ with } x > 1 \exists \text{ a } GU_{x-1,j} \text{ in its ancestor node that is the brief of } UU_{x,j} \tag{18}$$

According to (18), $GU_{x-1,j}$ is the group resultant of merging the whole set of user's groups in the node $UU_{x,j}$. This merging procedure is supported by the fusion function φ (11), the same already used in the former KLUM model to keep the knowledge base within a controlled size. In this case, the upper bounds (γ and η) restrict the number of group descriptions and information atoms (respectively) of a sub universe of users, that is, any node in the tree. The first exception is about η and the leaf nodes, where that bound should not be applied (to avoid knowledge loss, and because those nodes are much smaller anyway). But the main difference is that in this case the knowledge is not only summarized but distributed (and then summarized). When any node grows to exceed the upper bound on the size (γ), it will be partitioned into two (or more) nodes. The old node will be replaced by the new ones, having its elements distributed between those new nodes. The summary of the old node, located at its ancestor, will be replaced with summaries of the new nodes, each of them attached to the correspondent pointer to node.

After stating the general idea, let's review the whole insertion procedure. When a session ends, the *CUDS* (4) is stored as a new group of users with population 1. According to the definition of R-tree new insertions must be done in leaf nodes. For that purpose the most suitable leaf node must be found. First of all, *CUDS* is compared with all groups of users in the root node, using the match function M (5). The group description providing the best match with the *CUDS* will be chosen, obtaining a pointer to a node in the lower level. This procedure is repeated until a leaf node is reached, and there the new group will be inserted. Notice that the formulation for the match function (and its subsequent supporting functions) is the same that the one applied to the previously explained model subject to knowledge loss (KLUM).

When a leaf node exceeds the upper bound γ, the partition process is triggered. This procedure relies upon a distribution function ρ (19), which provides a distribution of the elements within a sub universe of users into several ones (at least two) by following some criteria (specific of each implementation).

$$\rho : UU_{x,y} \rightarrow \{UU_{x,z}\} \mid z > 1, \bigcup_z UU_{x,z} \equiv UU_{x,y} \quad \text{and}$$

$$\forall i,\ jUU_{x,i} \cap UU_{x,j} \equiv \varnothing \tag{19}$$

In order to prevent the tree to grow in depth, all the nodes should meet a minimum occupation requirement ($k_{\min}$). Given that an overflowed node has $\gamma + 1$ elements, the maximum cardinality of a sub universe resultant from partition will be $\gamma + 1 - k_{\min}$. In fact, this is a requisite applied by the specific implementation of ρ. In the following formulation (20), a proposal of such implementation is provided in order to illustrate the general definition of ρ.

$$\begin{aligned}
&\text{Let } \rho_0(UU_{x,y}, \varnothing, \varnothing) \\
&\quad \equiv \big(UU_{x,y} - \{GU_a, GU_b\}, \{GU_a\}, \{GU_b\}\big) \,\big|\, \\
&\quad GU_a, GU_b \in UU_{x,y} \wedge \forall GU_c, GU_d \in UU_{x,y} \\
&\quad \text{is } M(GU_a, GU_b) \leq M(GU_c, GU_d) \\
&\forall i \neq 0,\ \text{let } \rho_i(A, B, C) \equiv \big(A - \{GU_a\}, B', C'\big) \mid \\
&\quad GU_a \in A \wedge \forall GU_b \in A \text{ is } \max\big(M(GU_b, B), M(GU_b, C)\big) \\
&\quad \leq \max\big(M(GU_a, B), M(GU_a, C)\big) \\
&\text{with } \big(B' \equiv B \cup \{GU_a\} \wedge C' \equiv C\big) \\
&\quad \text{iff } |C| \geq \gamma + 1 - k_{\min} \vee \big(|B| < \gamma + 1 - k_{\min} \\
&\quad \wedge M(GU_a, B) \geq M(GU_a, C)\big) \\
&\text{and } \big(B' \equiv B \wedge C' \equiv C \cup \{GU_a\}\big) \text{ in other cases} \\
&\rho(UU_{x,y}) \equiv (UU_{x,1}, UU_{x,2}) \\
&\quad \leftrightarrow \rho_n\big(\rho_{n-1}\big(\ldots \rho_0(UU_{x,y}, \varnothing, \varnothing)\ldots\big)\big) \\
&\quad \equiv (\varnothing, UU_{x,1}, UU_{x,2}),\ n = |UU_{x,y}| - 2
\end{aligned} \tag{20}$$

After creating these two (or more) new nodes, their briefs are obtained as the fusion of all its content (all the elements within). These briefs with the correspondent pointers to the new nodes are inserted into the parent node. Finally, the overflowed node (and, consequently, its brief at the ancestor) is removed. This general rule can be applied to every node except from the root node. If the root node overflows, it will be partitioned just in the same way. But in such case a new root node must be created, containing the briefs and pointers to the new nodes.

Notice that by increasing the number of group descriptions in the ancestor, it may be overflowed in turn leading to a new partition process, and so on till the root. Even though multiple partitions would affect the system's performance, this is an infrequent scenario in R-trees. In fact, even regular partitions are infrequent due to the R-tree structure. Besides, inserts that don't trigger a partition may lead to obsolescence of its brief at the ancestor, given that the brief was obtained without taking into account this newer element. Because of this, updating that group description is required at the ancestor, and in fact also at every higher level till the root. Updating processes at every insert may involve a high computational cost. The number of updates can be reduced by controlling the number of changes (updating each x changes, instead of updating every time). Anyway, it should be pointed out that the eventual loss of performance due to inserts does not affect any critical process, since insertion is an offline process (it runs once the user is disconnected).

On the other hand, both precision and efficiency are critical in the inference process. Because of this, processing several nodes of the tree is usually unaffordable, even for an only branch of the tree. Instead of doing this, the inference process should focus on a unique node for efficient responses. Thus, the notion of a current window is added to the model. The current window is a pointer to node added to the *CUDS* (21). Initially, at the beginning of the session, it is set to the root node of the tree. As the *CUDS* are matched with the *GU*s within the root, the best match will be checked with the test function (τ). That function establishes when a match is good enough and trustworthy. The definition of the function and its nature may be diverse. It can be as simple as a certainty threshold, different thresholds for each level of the tree, a threshold for the difference between the best match and the second one, etc. Anyhow, when the match satisfies the test function then the pointer to current window will be updated to the node represented by the group providing that outstanding match. New matches will be performed in that node, and so on till a leaf node is reached.

The *CUDS* is a *UD*, its stereotype and the current window being i the root level at the beginning of every interaction

$$GU(CUDS) = (CUDS, 1, UU_{ij}) \tag{21}$$

Once the sub space of users is selected, and having the matches already calculated, the inference is accomplished by applying a selection function (12), (13), as in the former KLUM model.

Summarizing, R-tree based user model avoiding knowledge loss is formalized in RTUM($F, \gamma, \eta, M, \varphi, \iota, \rho, \tau$), keeping the same elements as in KLUM and with the same definition but adding two new functions due to the data structure it is based on: the distribution ρ, for partitioning overflowed nodes, and the match-test τ for updating the current window where inferences are performed. Again, although the model is defined by generality, examples of general functions are added to illustrate the proposal and to depict a specific implementation, which will be evaluated in the following section.

4 Evaluation

The following section provides an evaluation of both user model approaches proposed during this document. Seeking load diversity, several test domains are taken into account.

Table 3 Domain characterization

	Real & open domain	Artificial domain	Benchmark domain
Number of users	100000	100000	40000
Number of features	23	22	29
Maximum feature cardinality	177851	90196	53
Minimum feature cardinality	57	7520	2
Average feature cardinality	41788	31967	7
Maximum features per user	21	22	27
Average features per user	11	21	22
Minimum features per user	3	18	20

Firstly, the evaluation design will be briefly explained. Then, the experiment parameters and workload will be described. Finally, the results are presented divided in data sets (by different kinds of test domain) and discussed in a separate sub section.

4.1 Evaluation goals and design

The main purpose of this evaluation is checking the advantages of RTUM with respect to other models subject to knowledge loss. Because of this, a comparative evaluation has been performed keeping the same implementation for any function the models have in common: match (5), fusion (11) and inference. In the case of the inference function, two examples have been posed ((12) and (13)), and both will be evaluated. A third 'inference' function will be included for establishing a baseline. This new function just provides the most frequent value for the feature in its domain (default value), despite the knowledge gathered in the base. Regarding the additional functions required for RTUM, the distribution function proposed as an example (20) will be used in this evaluation. Finally, the match-test function (τ) will be implemented as a certainty threshold that will be acquired through preliminary experimentation (see Sect. 4.3).

The models are tested in three different domains: a real and open domain; a bounded machine-generated domain, in the same style as an open domain but with some features validated by ontology; and a real and bounded domain from a research project, whose data is validated by means of ontology. The later domain will be hereinafter referred as the research project it comes from, that is Cadooh (TSI-020302-2011-21) financed by the Spanish Ministry of Industry, Tourism and Trade. In addition, this third domain has been used to build a Benchmark for Statistical User Modeling. This Benchmark is a framework to evaluate and measure the performance of any user model for user modeling in order to make all evaluations replicable and comparable over different models. Main issues concerning the structure of the Benchmark data set will be described in Sect. 4.2, yet some of its features are described in the following.

Such different domains can provide a more general comparison of the approaches. As can be seen in Table 3 there are several differences among the three domains. The open real data domain has an average feature cardinality of 41788, which is extremely high as usual in open domains. Moreover, several users within this domain are described by just three rows per user. In sum, such domain types harden learning, and frequently the model does not have enough information about the user to provide reliable inferences. Thus, the results of this type are not expected to be good, but it will be included in the evaluation anyway in order to provide a more complete comparison of the described user models. On the other hand, the Cadooh domain has a more common characterization for the specific domain application type, which shapes a more meaningful part of the evaluation.

Regarding the sample characterization, three different sample structures exist, each one coming from one of the three domains (see Table 4). It is especially interesting to remark the percentage of multi-valued features on every sample, since high percentages of multi-valued features will negatively affect system reliability. The open domain has the highest percentage reaching an 81.8 % of multi-valued features, while the others have more usual values (24.2 % and 3.33 %, respectively).

In addition, the Benchmark domain belonging to Cadooh project has been divided into two different data sets, training set (28000 users) and test set (12000 users). Both data sets were used to build the Benchmark yielding the following tests. The training set experiment involves 3650 randomly chosen users setting a confidence level of 99 % and a confidence interval of 2 %. On the other hand, the test set experiment involves 3100 randomly chosen user setting a confidence level of 99 % and a confidence interval of 2 %. Remark that all evaluations accomplished over this benchmark involves fixing a random feature from a reduced set of three features that are considered the most frequently asked features during an interaction.

Regarding how evaluation is going to be accomplished, firstly the models are trained by feeding all the user descriptions. Then, a subset of the domain (2,500 user descrip-

Table 4 Samples characterization

	Real data domain	Artificial domain	Benchmark domain
Number of users	$2 \cdot 10^4$	$2 \cdot 10^4$	$28 \cdot 10^3$
Maximum rows per user	21	29	33
Minimum rows per user	3	21	18
Average rows per user	12	26	25
Maximum features per user	21	29	27
Minimum features per user	3	19	18
Average features per user	11	25	22
Percentage of multi-valued features per user	81.8 %	24.2 %	3.33 %
Percentage of multi-valued rows per user	83.3 %	37.8 %	5.42 %

tions for real data domain and artificial domain and 3100 for benchmark domain) will be chosen as a test set (the same for both models). Then two different ways of evaluation will be applied. The first one requires fixing a random feature that will be queried later, and the rest of the tuples will be fed randomly to refine the *CUDS* iteratively (the same random order for both models). For each iteration, the fixed feature is queried and the model's prediction checked with the real value. The second method is raised by a real application case, within the research project Cadooh. In this case, the queried feature is in a reduced set (of just nine features) and the rest of the features are fed to the model iteratively in a pre-fixed order, which is the order in which information atoms are acquired at almost every interaction in that interactive system. The latter method will only be applied to the third domain (the real and bounded domain, related to that research project). It is included since it is expected to provide information about how the order of the features affects system reliability. Anyhow, the model is queried iteratively as the *CUDS* evolves, because the model will probably provide more reliable inferences when it knows more about the current user and this should be checked.

Finally, regarding the metrics, the success rate is observed as the number of correct predictions divided by the number of users in the test set. The response times for the inference process will also be observed. It should be pointed out that the matching processes could be advanced in real systems as information about the current user is acquired. However, prediction requests may come immediately after acquiring information, when the match is not finished. Consequently, the response time observed will include the time required to perform the matching.

4.2 Framework to evaluate statistical user models

There are many works aimed to check how the user-adapted interaction systems improve the success of interaction and enriches the interactive experience of the users. Talking more specifically about this type of user models, empirical evaluations are needed to determine how much reliable is the statistical inference on the models built. However, evaluating a statistical user model is a non-standardized and time consumption task. For several years, according to [20], there is a lack of evaluation tools (methods and benchmarks) that has not been covered yet. The key to achieving high-quality empirical evaluations is in designing proper experiments that could be tested separately.

The benchmark presented here includes a database of users (split in a training set and several test sets) and sequences of experiments to perform the evaluation of a user model of this kind. This section describes main issues concerning the benchmark together with a statistical significance analysis in order to justify the size of each data set provided.

Benchmark corpus has been obtained from a real domain used in Cadooh (TSI-020302-2011-21) research project, where 4×10^4 samples (user descriptions) were acquired. Even though some of the features were biometric, the data domains of the user features were discretized in order to ease its application to any statistical user model. Main issues concerning these user descriptions can be found in Sect. 4.1. As aforementioned, the samples were split into a training set (28,000 randomly chosen samples) for feeding the statistical user model before its evaluation, and a test set containing enough samples (the remaining 12,000 samples) as to support tests with different confidence levels. The following Fig. 1 depicts main issues concerning benchmark samples including cardinality distribution, features per user distribution and rows per user.

The benchmark is completed with a series of experiments on the former data sets. The prototypical experiment involves taking a sample, which will serve as the 'current user', and selecting one of its facts (regarding a user feature) as the inference goal. The remaining facts of the sample will be iteratively fed to the already-trained model. For each iteration, the model will be requested to infer the selected fact for the current user. The main proposed metrics are:

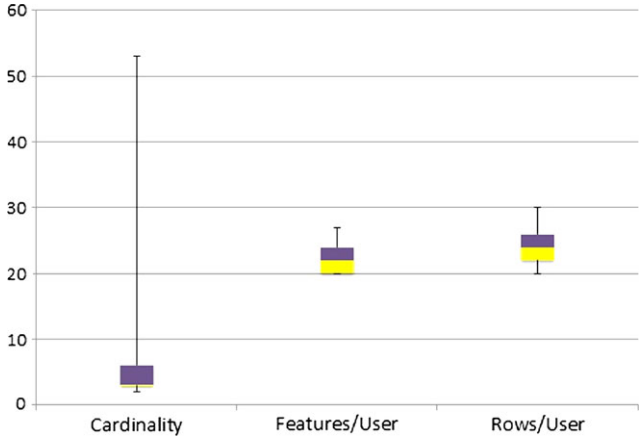

Fig. 1 Benchmark structure

(a) the inference success (if the model infers the correct fact);
(b) the number of iterations required to get the correct fact;
(c) the certainty of successful inferences;
(d) the certainty of unsuccessful inferences.

To preserve the validity of the results through metric (b), the model should not be fed back with the results of its inferences (either success or fail). Thus, at next iteration the model can infer again the same (unsuccessful) value unless the newly fed fact about current user makes it to change its consideration. Nevertheless, another method and metric can be posed by feeding back the model and checking how many iterations requires to success. Finally, additional metrics are added to test the efficiency (time consumption) while comparative evaluations based on this metric should observe using similar hardware resources for running the compared user models.

The batteries of experiments should be instructed both within the training set and out of it (that is, in the test set). For building them, the appropriate analysis of statistical significance was performed. The goal was to ascertain the sizes of each battery, provided different confidence levels and intervals. For example, with the confidence level set to 99 % and the confidence interval set to 2 %, it was obtained that 3650 samples within the training set and 3100 samples of the test set were required.

This benchmark (http://labda.inf.uc3m.es/doku.php?id=en:labda_lineas:um_1) is for public usage. In that webpage there is also a detailed description and some examples of use.

4.3 Experiments preparation

Both proposals are developed in Java and PL/Sql. The knowledge bases are supported by relational databases DBMS Oracle 11g. In addition, both proposals have a set of Java methods embedded in its knowledge base. The experiments have been launched using the shell Cognos.User [19]. To perform the evaluation two SUN XFire with Intel Xeon E5450 at 3 GHz, 8 GB RAM and SAS hard disc (4 ms latency) are available. Client machines are Intel Pentium 4 at 3 GHz with 2 GB RAM running Windows 7 with Java RE v1.6 installed. According to these hardware specifications and model parameterizations both model's response times are set to be lower than one second (a requirement of the research project where the model was developed). In order to meet this requirement some model's parameters will be set. The upper bounds for the size of the knowledge base keeping the performance requirements depended upon the implementation and Hw resources, and thus had to be ascertained through several preliminary experiments. The bounds were averaged, then rounded and finally set to $\gamma = 50$ group descriptions and $\eta = 8000$ information atoms.

Besides, the RTUM model will keep the same bounds γ and η, yet it provided predictions quicker than the KLUM in those preliminary experiments. Thus, their response time will be compared later. Finally, some preliminary experimentation was also run for fixing the function match-test (τ). With a random set of 500 users, the clustering was checked searching for local maximums in relation to success and certainty of the match. A certainty threshold of 0.54 was chosen for τ, that is the function that establishes the match is good enough if it is equal to or higher than that threshold.

4.4 Evaluation in open interaction domain

The first part of the evaluation involves testing both user models working on an open interaction domain. Yet this sort of domain is not the current goal for these models, usually aimed to closed and reduced domains for specific interaction systems, it was considered interesting to include this part of the evaluation to gain perspective about the behavior of the models.

As mentioned before, this sort of domain observes multi-valued features and contains many users with few rows. These are the main drawbacks of this sample and both user models must deal with these disadvantages when performing the evaluation. All these issues affect system performance and reliability and likewise the success rates.

For this evaluation, a workload of real users was available. However, this dataset was not ontology-validated, adding the drawback of different terms (words) referring the same concept. Because of this, a Machine-Generated ontology-validated dataset of this kind was generated to complete this sort of evaluation. The following subsections provide the results on each of these datasets.

Table 5 Open domain results

Open domain	Maximum fit	Fit and certainty	Default value
KLUM	12 %	14.7 %	0.01 %
RTUM	14.3 %	15.02 %	0.01 %

Table 6 Artificial domain results

Artificial domain	Maximum fit	Fit and certainty	Default value
KLUM	18.68 %	18.72 %	0.1 %
RTUM	20.31 %	21.89 %	0.1 %

4.4.1 Open interaction domain: real dataset

A brief of the evaluation results is shown in Table 5. Regarding KLUM, the maximum fit method yields a success rate of 12 % while the method weighting fit values and certainty values produces a success rate of 14.7 %. First, it should be highlighted that in this dataset the success rate of the *default value method* is 0.01 %. The model RTUM using the *maximum fit method* provides a success rate of 14.3 % while the method aggregating *fit and certainty* yields a success rate of 15.02 %. Even though a preliminary analysis might show off that success rates are pretty low, it should be reminded that some of the domain characteristics have a very high percentage of multi-valued features (81.8 %), non-validated values (free writing), and under-described users (with few atoms).

4.4.2 Open interaction domain: machine-generated dataset

The purpose of this dataset, the same type as that of the preceding section, is to avoid the multiplicity of terms for the same concept produced by a 'free writing' interaction style. Thus, it has high values for both the average feature cardinality and percentage of multi-valued features.

In this dataset, further described in Sect. 4.1, the success rate of the *default value method* is 0.1 %. The KLUM yields a success rate of 18.68 % using the *maximum fit method* and an 18.72 % when performing the *fit and certainty method*. Regarding RTUM, it yields a success rate of 20.31 % when performing the *maximum fit method* and 21.89 % when applying *fit and certainty*. These results are briefed in Table 6. It can be observed that for this domain, success rates are higher than the ones obtained in open domain and that success rates are pretty higher than domain default value.

4.5 Evaluation on benchmark

Most significant part of the evaluation is that regarding real cases of application on closed interaction domains for specific systems, in which this type of model has proven satisfactory performance providing significant benefits to the interaction. For making the evaluation replicable and comparable with other models, it has been developed on a benchmark available for public use and free of cost, already described in Sect. 4.2.

The evaluation includes experiments on samples both within and outside the training set. On the other hand, the evaluation required to observe the restriction of a specific implantation of the model (Cadooh), where the facts were acquired in a predetermined order (certain user features are early acquired in a given order) and the data to be inferred was restricted to a subset of three queries (regarding user preferences and future behavior). Nevertheless, the evaluation will include both experiments subject to these constraints and experiments of general type (any feature to be inferred and facts fed in random order). Consequently, the results are organized into next four subsections by combining the two classification criteria.

4.5.1 Evaluation on benchmark: within training set by random order

The results of this first battery of experiments on the benchmark are summarized in Fig. 2. In this case, the *maximum fit method* performs better than the *fit and certainty method* with both models. With KLUM, their success rates are 62.87 % and 41,78 % respectively, while the *default value method* rises to 16 %. This latter method performance depends on the dataset (and not on the model), and therefore this rate will be constant for all experiments within the training set. Regarding the RTUM performance, the success rates reach 70.10 % and 67.15 % respectively, thus beating both methods of fit and certainty and default value.

4.5.2 Evaluation on benchmark: within training set following a pre-fixed order

The results obtained during this experiment are displayed in Fig. 3. In this case, the performance of both methods *maximum fit* and *fit and certainty* are very similar. In fact, the average rates obtained by the *fit and certainty method* are just like in the previous case with both models. However, the averages obtained through the maximum fit drop to align with the behavior of the other method.

With KLUM, the fit and certainty method yields a success rate of 41.77 % (barely lower than the 41.78 % obtained with random order), while the maximum fit method does not get more than 41.36 %. The RTUM, in turn, provides a success rate of 67.15 % with the fit and certainty method, and slightly higher 67.47 % with the maximum fit method.

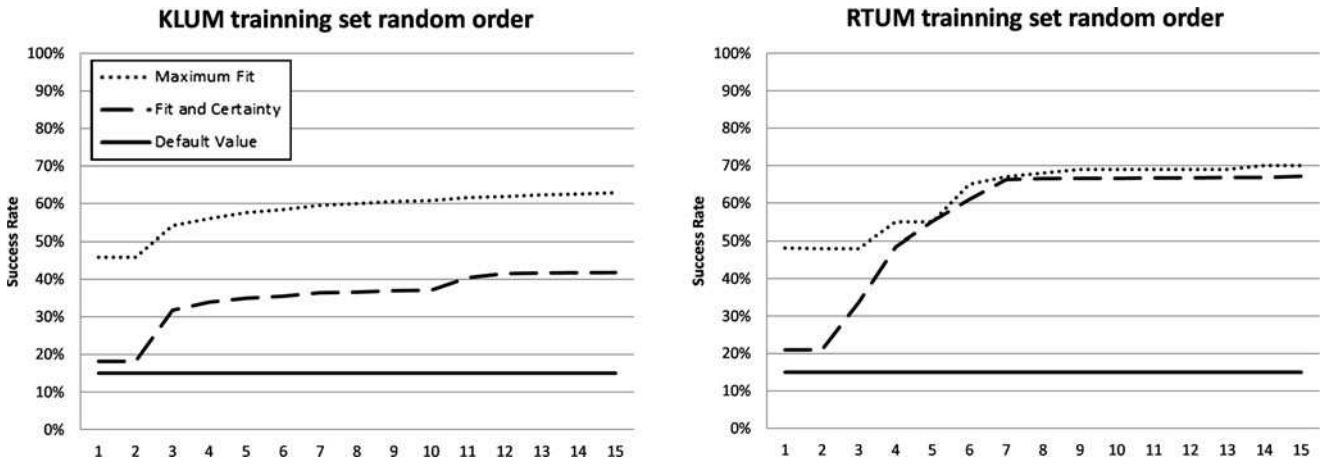

Fig. 2 Success rates of both models on samples in the training set which facts are fed in random order

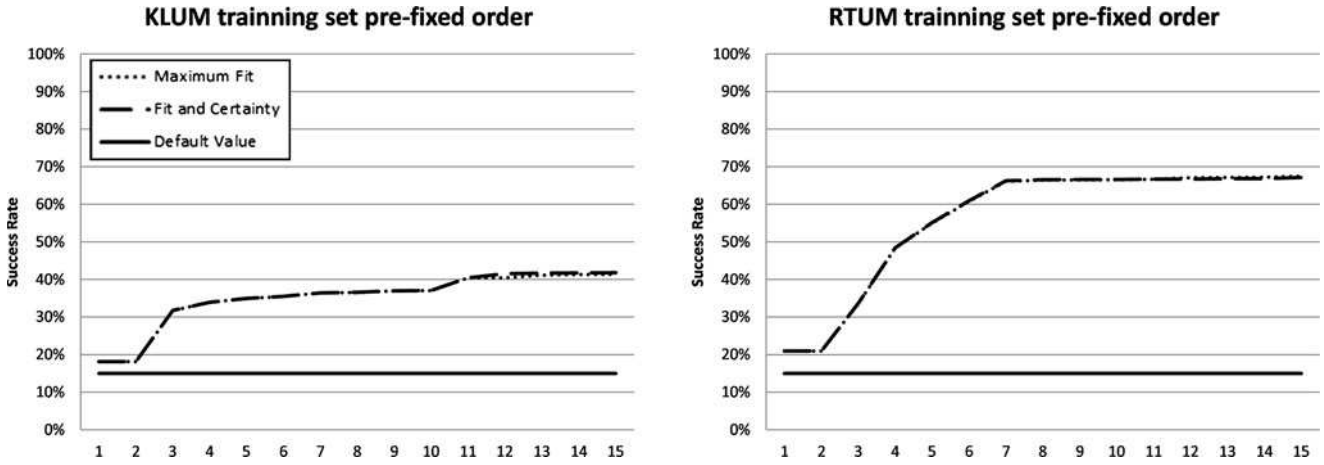

Fig. 3 Success rates of both models on samples in the training set which facts are fed in pre-fixed order

4.5.3 Evaluation on benchmark: in test dataset by random order

The samples used for these experiments have not been involved in the training process. The benchmark includes different batteries of experiments calculated to be statistically significant for different values of the confidence level and interval. In this case, the experiments run are supported by the workload prepared with a confidence level of 99 % and within a confidence interval of 2 %.

Figure 4 gathers mean results of both models when tested with that workload, and following the procedure of feeding the facts of the samples by random order. Once again the higher success rates are obtained by RTUM and the method of *maximum fit*. KLUM yields average success rates of 63.25 % (maximum fit) and 30.61 % (fit and certainty), while RTUM gets 72.45 % and 46.90 % (respectively).

4.5.4 Evaluation on benchmark: in test dataset following a pre-fixed order

Figure 5 shows the results obtained by both models performing over the benchmark test set and using a predetermined order of the facts to be fed. This order is, of course, is the same already applied in the experiments of the same kind on the training set. The results follow the same pattern as that depicted by the experiments within the training set with pre-fixed order: the fit and certainty method keeps its average success rates, while the maximum fit method drops its performance down to the levels of the other method. Specifically, KLUM shows average rates of 29.67 % and 30.61 % respectively, while RTUM provides 46.90 % and 47.90 % respectively.

4.6 Results regarding efficiency

Performance measurements are pretty important taking into account that this type of systems must perform in bounded

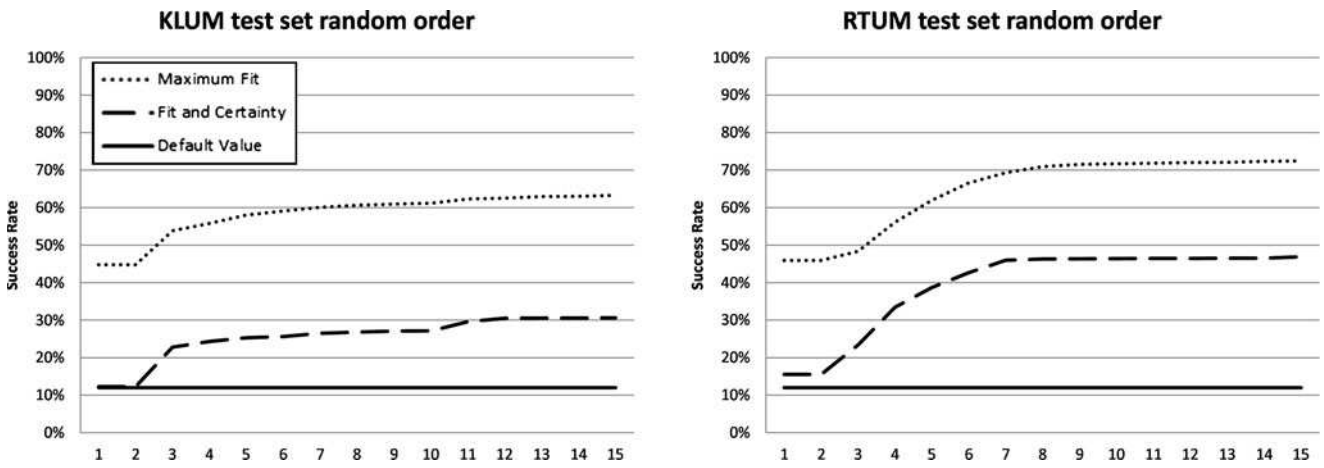

Fig. 4 Success rates of both models on samples in the test dataset which facts are fed in random order

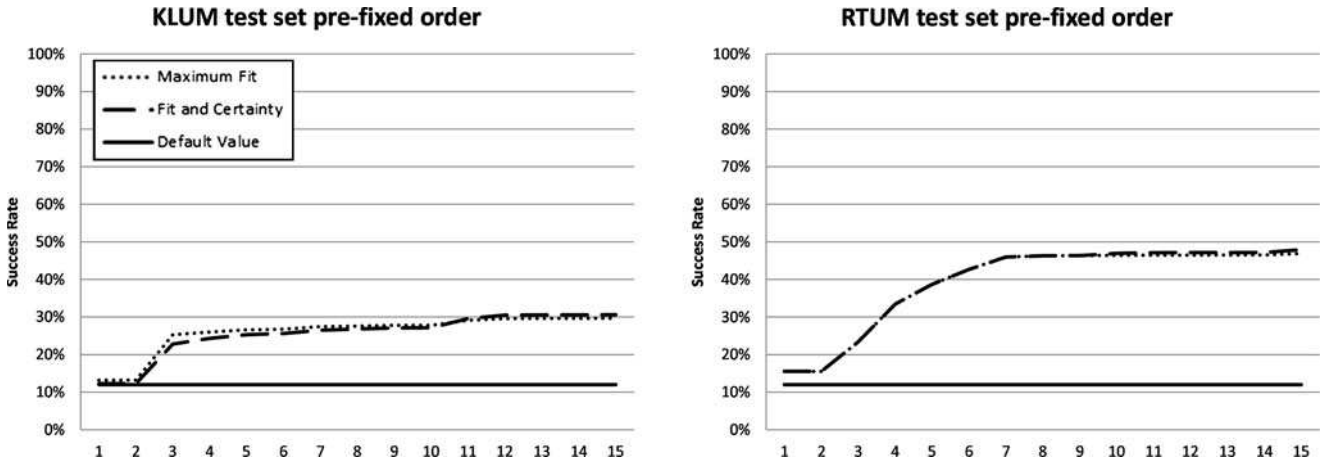

Fig. 5 Success rates of both models on samples in the test dataset which facts are fed in pre-fixed order

time. Most critical process, regarding response time, is the inference since it is performed during interaction (on-line process). Through these experiments, time measurements yielded that RTUM is on average a 30 % faster than the KLUM. In fact several time measures during the experiments yields that RTUM takes 0.73 seconds on average to achieve one inference while KLUM takes nearly one second to perform the same operation. Therefore, it can be concluded that RTUM not only shows better success rates obtained, but also performs faster than this model.

As stated in Sect. 3, the success rates of RTUM can be boosted by increasing the upper bound γ. However, such procedure causes a severe performance drop. Therefore, that upper bound should be established for each implementation by observing the response time requirements and the available hardware capabilities, thus balancing performance and efficiency. Another way of increasing the success rate is to eventually observe several *inference windows* (when two branches are found to be seemingly good, both nodes can be processed in parallel). However, this solution boosts the computational cost and should only be applied where the hardware resources are high enough or the response time constraints are relaxed.

Although the cost of maintenance is not critical because it runs off-line, it is important to reduce it as much as possible to prevent the machine from becoming overloaded. The KLUM model requires a maintenance operation after every session (when acquiring new knowledge). Furthermore, the cost of this operation increases as the knowledge base grows (with the use of the system). To keep the base volume (and hence the maintenance cost) within reasonable margins, upper bounds were set for the size of the base (maximum number of facts), thus stabilizing the cost. Through this experimentation, the average time consumption of closing a session in KLUM is 776 milliseconds, showing a standard deviation of 36 milliseconds.

In turn, the RTUM requires to run a similar maintenance process on the leaves every $\gamma/2$ sessions held on average (25 in this experimentation). In addition, non-leaf nodes of an updated branch also need frequent updates, but since they in-

volve minor changes they can be run periodically every few sessions (five sessions, in these experiments). Thus, closing a session in RTUM takes on average 180 milliseconds, showing a standard deviation of 708 milliseconds (usually takes 35 milliseconds, but could take up to 3.65 seconds). As a consequence, the training processes are harder in KLUM (during this evaluation, RTUM's training took less than a quarter of the time required for training the KLUM).

4.7 Discussion

As a consequence of the evaluation results, it can be stated that the RTUM performs better than KLUM in every evaluated scenario. When evaluating on Benchmark, better success rates were obtained using the maximum fit method (reaching a success rate of 72.45 % in real system conditions). On the other hand, the fit and certainty method behaves better in open interaction domains (where the features to be inferred are defined on excessively extent domains).

However, both methods yield similar results when considering a pre-fixed order to feed the features. Deeper analysis of this result leads to realize that in these cases the match of current user with an only stored group is stronger (despite of the correctness of this match). Thus, most probable value in the most probable group (*maximum fit method*) has a relatively high certainty value. The fit with the rest of the groups within that universe of users are much lower, and the certainty of any of those alternative fits multiplied by the certainty of any of their candidate values will get a negligible value. Because of this, the value maximizing the summation of the products of group fit degree with the certainty of the value in that group (*fit and certainty method*) coincides with that most probable value in the most probable group. In sum, both methods infer the same value in almost all the cases, and consequently they have very similar success rates.

Anyhow, the wrong matches are the cases to focus in order to improve the rates. In further analysis of those cases, it can be seen that in those cases the facts fed regarded features non discriminant between the different user groups stored in the node. Observing this conclusion through the maintenance processes would improve the structure of the knowledge and the success rates of inferences on it. In the KLUM, by privileging more frequent features through the fusion process which will force such features to be more discriminant (thus reducing the cases of uncertainty to infrequent combinations of facts). And in the RTUM by privileging those features through the partition process, which will make the tree-like structure to produce clearer matches in more cases (thus improving the success rates). These conclusions should be taken into account for further work, in order to explore the room for improvement of the models.

It should be pointed out that RTUM goes to more specific knowledge when current window focuses on lower levels in the tree. Eventually, it may mistake the chosen branch when descending to a lower level (especially when it comes to choose a leaf node, since the decision is harder at lower levels). However, it should be pointed out that in such cases the inferences keep a good tone, because the chosen branch is similar enough to the proper branch. Anyhow, if further definition of current user is provided (that is, new information about him is acquired) it would be of interest not only going deeper in the three-like structure, but also checking former decision (that could be obsolete because of new information). In sum, a line of further improvement is directed to periodically check if the current inferences window is appropriate, and if it is not change it back to focus the ancestor node.

5 Concluding remarks

The proposed RTUM model improves the performance of other models subject to knowledge loss, such as the KLUM. Since both approaches have been tested with the same formulation, the improvements are mainly due to its R-tree structure. As shown in the evaluation, RTUM has obtained better results in all the evaluations performed, both in success rates and in the response time.

RTUM is able to perform with features that cannot be modeled by domain experts. For example, the selection of different dialogue strategies adapted to users is hard to model but can be easier supported by knowledge extracted from experience by essaying strategies and checking the consequences (following a *trial and error* method). This typology of features has been successfully tested in Cadooh domain, providing further adaptation. In that case, the model received a negative feedback after applying an unsuccessful strategy, which was learned and discouraged later selection of this strategy for similar users. The essay of new strategies was decided when no value was possibly inferred for this feature. In such case, the available knowledge about this feature states which strategy is not appropriate, but tells nothing about which strategy should be chosen. The proposed value (strategy) will be randomly chosen within the domain of the feature (except for the given inappropriate values). Consideration should be given, as a line for future work, to alternatives that converge on the most appropriate solutions in less time (for example, applying evolutionary computing).

Besides, both KLUM and RTUM models are prepared to acquire stereotypes provided by experts, thus alleviating the start-up problem (operation in the early stages of implantation, when experience is insufficient to draw accurate inferences). Pre-defined stereotypes will be stored as individuals with a population higher than one. As real sessions are held, these stereotypes will be reinforced if similar users interact, or dissipated if no similar users are found (the larger the population assigned, the longer the influence will last).

KLUM could even eliminate the wrong stereotypes. RTUM will never eliminate them, yet it will reduce its influence down to negligible levels. If users of such types come later, the stereotype will be progressively restored. For that reason, it can be concluded that RTUM is not only a group user model, but also a hybrid proposal that can perform with predefined (proposed by expert) stereotypes. Furthermore, it is here recommended to include those predefined stereotypes as the initial workload thus avoiding anomalous interactions in the initial period of operation, in which decisions are otherwise unsubstantiated because the knowledge base has insufficient content.

It should also be pointed out that the presented models preserve the anonymity of the users. Taking into account that there are no identifying features, several sessions of the same user will be stored as different individuals. Such an approach improves the matching of the current user taking into account previously held sessions, even more when considering dynamic features such as the frame of mind. On the other hand, both models have been presented taking into account the collaborative learning approach. It would be relevant to explore its application under the content-based approach even when in this case due to the reduced volume of sessions the KLUM model might provide better results.

The main disadvantage of RTUM is the amount of redundant knowledge stored in its knowledge base. As explained before, in this proposal the universe of users consists of the conjunction of all users in the leaves of the tree, the rest of the nodes being redundant knowledge. However, the possibility of excessive growth of the knowledge base is unlikely due to the minimum occupation per node. In fact, having one billion user sessions stored in the knowledge base and a minimum occupation per node of 30 %, the tree would have between 7 and 9 levels, which are still enclosed numbers of levels taking into account the number of users in the knowledge base. On the other hand, with a parameterization of the model similar to the presented one, that implementation could have up to $7 \cdot 10^7$ tables. Even with lower volume requirements, it is strongly recommended to count on a powerful database management system, capable of efficiently managing such data volumes and with many scalability possibilities. Focusing simultaneously on several nodes during the inference can improve the results. In addition, it should be pointed out that the RTUM model behaves better as the current window evolves, that is as nodes of lower level in the tree are focused. In order to avoid performance drop, parallel processing of these nodes should be observed. Thus, a good data management supporting data sharing and parallelism is necessary. Such database support raises the need of studying the database instance adaptation to the problem (physical data structures, cache policies, etc.) in order to improve performance, which will be left for later research. Better performance enables to set higher upper bounds γ and η, which reduces performance but may be still within the response time requirements. Notice, however, that the results shown in this paper have been obtained with a default set up of the database instance (with no tuning) and that they could be therefore improved.

As shown in the evaluation section, RTUM with the proposed formulation for its different functions behaves well (at least, for the evaluated cases). Both formulation and reasoning-mechanisms have been designed to be general and applicable to any domain. However, the effect of variations on those formulas through different domains should be examined, in order to study the proper formulation for each interaction domain. In this line, there are several works in the bibliography applicable to the partition formulation design. For example, most clustering solutions like K-means [49] or C-means algorithms [7, 48] can be adapted to explore alternatives and eventually enhance the proposed solution for the partition process in specific domains. There also can be found solutions suited to high-dimensionality data [36]. On the other hand, there also can be found alternatives substituting either the fusion procedure (such as computing centroids), or the matching function [2] during inferences and/or insertions. Besides, self-adaptive formulation is seen as an appealing challenge for further work. In this line, features provided in the early stages of interaction can be privileged in partition processes, thus improving discrimination at higher-level nodes. Privileged features can be either fixed in design time or discovered through lifetime (self-adaptation). That approach mitigates also the cold-start problem (difficulties in classifying user at the early stages of interaction). Biometric-type features (provided by cameras, microphone, or special devices) are a good example of items available from the beginning of the interaction.

In another vein, feature relevance is domain dependent, and this fact should affect formulation (weighting them properly through the matching process, for example). As with the privileged features, feature relevance can be modeled by experts or learned by the model instance through its use during its lifetime. The authors are currently working along these lines with Ontology support in order to improve inferences accuracy and to ease the implantation of the model onto new interaction domains, enhancing reusability.

In fact, the model implementation is completely domain independent. However, the knowledge bases, which are built through experience during system's lifetime, are suited to the implantation domain. To reuse knowledge bases, or to enable an only implantation providing services for several domains, it is necessary to meet the following two requisites: the involved universes of users are not disjoint; and there is a mechanism to adapt user characterization to the peculiarities of each domain. In this later issue, the mentioned observation of feature relevance can make a difference. With these developments, authors consider that the model could

be launched as an independent decision-support system (in its current exploitation, it is a component in the tourist information system Cadooh).

Finally, a benchmark proposal has been developed in order to provide current State of the Art with a framework to measure and compare the performance of statistical user models. This public and free-of-use benchmark includes two datasets (for training and test) and several batteries of experiments according to diverse combinations of the parameters of the statistical significance analysis.

Acknowledgements This proposal development belongs to the research projects Thuban (TIN2008-02711), MA2VICMR (S2009/TIC-1542) and Cadooh (TSI-020302-2011-21), supported respectively by the Spanish Ministry of Education and the Spanish Ministry of Industry, Tourism and Commerce.

References

1. Adomavicius G, Tuzhilin A (2007) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
2. Ahn JJ, Byun HW, Oh KJ, Kim TY (2012) Bayesian forecaster using class-based optimization. Appl Artif Intell 36(3):553–563. doi:10.1007/s10489-011-0275-2
3. Albrecht D, Zukerman I (2007) Introduction to the special issue on statistical and probabilistic methods for user modeling. In: User modeling and user-adapted interaction
4. Amento B, Terveen L, Hix D, Ju P (1999) An empirical evaluation of user interfaces for topic management of web sites. In: Proceedings of the conference on human factors in computing systems (CHI '99). ACM, New York, pp 552–559
5. Badi R, Bae S, Moore JM, Meintanis K, Zacchi A, Hsieh H, Shipman F, Marshall CC (2006) Recognizing user interest and document value from reading and organizing activities in document triage. In: Proceedings of the 11th int. conference on intelligent user interfaces (IUI). ACM, New York, pp 218–225
6. Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley, Reading
7. Bahrampour S, Moshiri B, Salahshoor K (2011) Weighted and constrained possibilistic C-means clustering for online fault detection and isolation. Appl Artif Intell 35(2):269–284. doi:10.1007/s10489-010-0219-2
8. Baraglia R, Silvestri F (2007) Dynamic personalization of web sites without user intervention. Commun ACM 50(2):63–67
9. Basu C, Hirsh H, Cohen W (1998) Recommendation as classification: using social and content-based information in recommendation. In: Proceedings of the 15th national conference on artificial intelligence, Madison, WI, pp 714–720
10. Baus J, Kruger A, Wahlster W (2002) A resource-adaptive mobile navigation system. In: Proceedings of the 7th international conference on intelligent user interfaces (IUI). ACM, New York, pp 15–22
11. Bessa A, Laender AHF, Veloso A, Ziviani N (2012) Alleviating the sparsity problem in recommender systems by exploring underlying user communities
12. Billsus D, Pazzani M (1998) Learning collaborative information filters. In: Procs. of the 15th international conference on machine. Learning (ICML '98)
13. Billsus D, Pazzani M (2000) User modeling for adaptive news access. User Model User-Adapt Interact 10(2–3):147–180
14. Billsus D, Brunk CA, Evans C, Gladish B, Pazzani M (2002) Adaptive interfaces for ubiquitous web access. Commun ACM 45(5):34–38
15. Birukov A, Blanzieri E, Giorgini P (2005) Implicit: a recommender system that uses implicit knowledge to produce suggestions. In: Proceedings of the workshop on multi-agent information retrieval and recommender systems at the 19th international joint conference on artificial intelligence (IJCAI)
16. Breese J, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative clustering. In: Proc. of the fourteenth annual conference on uncertainty in artificial intelligence, San Francisco, p 43
17. Burke R (2002) Hybrid recommender systems: survey and experiments. User Model User-Adapt Interact 12(4):331–370. doi:10.1023/A:1021240730564
18. Calle FJ, Albacete E, Olaziregi G, Sánchez E, Valle Agudo D, Rivero J, Cuadra D (2011) Cognos: a pragmatic annotation toolkit for the acquisition of natural interaction knowledge. In: 27th conference of the Spanish society for natural language processing, Huelva (Spain)
19. Castaño L, Calle FJ, Cuadra D, Castro E (2011) User modeling for human-like interaction. In: The 2nd international workshop on user modeling and adaptation for daily routines (UMADR), pp 23–34
20. Chin D (2001) Empirical evaluation of user models and user-adapted systems. In: User modeling and user-adapted interaction, pp 181–194
21. Cosley D, Lam SK, Albert I, Konstan JA, Riedl J (2003) Is seeing believing? How recommender interfaces affect users' opinions. CHI Lett 5
22. De Meo P, Quattrone G, Rosaci D, Ursino F (2012) Bilateral semantic negotiation: a decentralized approach to ontology enrichment in open multi-agent systems. Int J Data Mining Model Manag 4(1):1–38
23. Eyharabide V, Amandi A (2012) Ontology-based user profile learning. Appl Artif Intell 36(4):857–869. doi:10.1007/s10489-011-0301-4
24. Finin T, Drager D (1986) GUMS: a general user modeling system. In: Proceedings of the workshop on strategic computing natural language. association for computational linguistics, Stroudsburg, PA, USA, pp 224–230
25. Formoso V, Fernández D, Cacheda F, Carneiro V (2012) Using profile expansion techniques to alleviate the new user problem. In: Information processing and management, ISSN 0306-4573
26. Ghazanfar MA, Prugel-Bennett A (2011) Fulfilling the needs of gray-sheep users in recommender systems, a clustering solution. In: International conference on information systems and computational intelligence, Harbin, China, pp 18–20
27. Guttman A (1984) R-trees: a dynamic index structure for spatial searching. In: ACM SIGMOD int. conference on management of data. ACM, New York, pp 47–57
28. Heckerman D, Chickering DM, Meek C, Rounthwaite R, Kadie C (2000) Dependency networks for inference, collaborative filtering, and data visualization. J Mach Learn Res 1:49–75
29. Herlocker JL, Konstan JA, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf Retr 5:287–310
30. Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53
31. Hill W, Stead L, Rosenstein M, Furnas GW (1995) Recommending and evaluating choices in a virtual community of use. In: Proceedings of ACM CHI'95 conference on human factors in computing systems. ACM, New York, pp 194–201
32. Krulwich B (1997) Lifestyle finder: intelligent user profiling using large-scale demographic data. Artif Intell Mag 18(2):37–45

33. Larson HJ (1969) Introduction to probability theory and statistical inference. Wiley, New York
34. Linden G, Smith B, York J (2003) Amazon.com Recommendations: item-to-Item collaborative filtering. IEEE Internet Comput, Jan–Feb
35. Miller B, Albert I, Lam SK, Konstan J, Riedl J (2003) MovieLens unplugged: experiences with a recommender system on four mobile devices. In: Proceedings of the 17th annual human-computer interaction conference (HCI 2003), British HCI Group, September
36. Özyer T, Alhajj R (2009) Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer. Appl Artif Intell 31(3):318–331. doi:10.1007/s10489-008-0129-8
37. Peddy CC, Armentrout D (2003) Building solutions with Microsoft commerce server 2002. Microsoft Press
38. Perrault CR, Allen JF, Cohen PR (1978) Speech acts as a basis for understanding dialogue coherence. Report 78-5, Department of Computer Science, University of Toronto, Canada
39. Popescul A, Ungar LH, Pennock DM, Lawrence S (2001) Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In: Proceedings of the 17th conference in uncertainty in artificial intelligence (UAI '01), pp 437–444
40. Resnick P, Iakovou N, Suchak M, Bergstrom P, Riedl J (1994) GroupLens: an open architecture for collaborative filtering of net news. In: Furuta R, Neuwirth C (eds) Proceedings of the 1994 conference on computer. supported collaborative work. ACM, New York, pp 175–186
41. Rich E (1979) User modeling via stereotypes. Cogn Sci 3(4):329–354
42. Rosaci D (2007) CILIOS connectionist inductive learning and inter-ontology similarities for recommending information agents. Inf Syst 32(6):793–825
43. Rosaci D, Sarnè GML, Garruzzo S (2009) MUADDIB: a distributed recommender system supporting device adaptivity. ACM Trans Inf Syst. 27(4):24
44. Rosaci D, Sarnè GML (2012) A multi-agent recommender system for supporting device adaptivity in e-commerce. J Intell Inf Syst 38(2):393–418
45. Sarwar BM, Karypis G, Konstan JA, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web (WWW '01), pp 285–295
46. Shardanand U, Maes P (1995) Social information filtering: algorithms for automating word of mouth. In: Proceedings of ACM CHI'95 conference on human factors in computing systems. ACM, New York, pp 210–217
47. Sheth B, Maes P (1993) Evolving agents for personalized information filtering. In: Proceedings of the 9th IEEE conference on artificial intelligence for applications
48. Siriporn S, Hwajoon K (2009) Modified fuzzy ants cluster-ing approach. Appl Artif Intell 31(2):122–134. doi:10.1007/s10489-008-0117-z
49. Tsai CF, Yeh HF, Chang JF, Liu NH (2010) PHD: an efficient data clustering scheme using partition space technique for knowledge discovery in large databases. Appl Artif Intell 33(1):39–53. doi:10.1007/s10489-010-0239-y
50. Ting CY, Phon-Amnuaisuk S (2012) Properties of Bayesian student model for INQ PRO. Appl Artif Intell 36(2):391–406. doi:10.1007/s10489-010-0267-7
51. Wade W (2003) A grocery cart that holds bread, butter and preferences. NY Times, Jan 16, 2003
52. Yu K, Schwaighofer A, Tresp V, Xu X, Kriegel HP (2004) Probabilistic memory-based collaborative filtering. IEEE Trans Knowl Data Eng 16(1):56–69
53. Yuan W, Guan D, Lee YK, Lee S (2011) The small-world trust network. Appl Artif Intell 35(3):399–410. doi:10.1007/s10489-010-0230-7
54. Zhang Y, Callan J, Minka T (2002) Novelty and redundancy detection in adaptive filtering. In: Proceedings of the 25th annual international ACM SIGIR conference, pp 81–88
55. Zukerman I, Albretch W (2001) Predictive statistical models for user modeling. User Model User-Adapt Interact 11:1–2

Javier Calle got his degree in Computer Science from the Technical University of Madrid in 1997. In 2000, he joined the Advanced Databases Group in the Computer Science Department at the Carlos III University of Madrid, where he is currently working as associate professor. In 2005, he obtained the Ph.D. degree in Computer Science from Technical University of Madrid, with a thesis in the Natural Interaction research. Apart from this, his research topics are focused on dialogue modeling, intentional processing and joint action models, and cognitive components of the interaction. He has also been working in several European and National research projects regarding Human-Computer Interaction.

Leonardo Castaño obtained his degree in Computer Science from Carlos III University of Madrid in 2010. In 2011 he got his M.Sc. in Computer Science from Carlos III University of Madrid. In 2007, he joined the Advance Databases Group in the Computer Science Department of Carlos III University of Madrid, where he is currently working doing his Ph.D. in Statistical User Modeling. His research interests include Human like interaction, Statistical User Modeling and Collaborative User Modeling. He has been working for nearly fourth months in the Intelligent User Interfaces Group at German Research Center for Artificial Intelligence (DFKI, Saarbrucken).

Elena Castro received the M.Sc. in Mathematics from Universidad Complutense of Madrid in 1995. In 1997, she joined the Advanced Databases Group, at the Computer Science Department of Carlos III University of Madrid, where she currently works as associate professor. In 2003, she obtained the Ph.D. degree in Computer Science from Carlos III University of Madrid. Her research interests include advanced database technologies, spatio-temporal databases and their applications to Situation management. She has been working in Computer Science Department at Purdue University of West Lafayette (Indiana) for nearly a year, where she has applied her research in Spatio-Temporal database.

Dolores Cuadra received the M.Sc. in Mathematics from Universidad Complutense of Madrid in 1995. Since 1998 she has been working at the Advanced Databases Group in the Computer Science Department at Universidad Carlos III of Madrid. In 2004, she obtained the Ph.D. degree in Information Science from Universidad Carlos III of Madrid. She is currently teaching Relational and Object Oriented Databases. Her research interests include database conceptual and logical modeling, Advanced Database CASE environments, Information and Knowledge Engineering, Natural Language Processing and E-learning field.