Knowledge-Based Systems 112 (2016) 114–126

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A weighted adaptation method on learning user preference profile

Zhiyuan Zhang[a,b], Yun Liu[a,b,*], Guandong Xu[c], Haiqiang Chen[d]

[a] School of Communication and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
[b] Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, Beijing 100044, China
[c] Advanced Analytics Institute, University of Technology Sydney, Australia.
[d] China Information Technology Security Evaluation Center, Beijing, 100085, China

ARTICLE INFO

Article history:
Received 14 April 2016
Revised 29 July 2016
Accepted 4 September 2016
Available online 10 September 2016

Keywords:
Recommender system
Numerical attributes
User preference profile
Dynamic

ABSTRACT

Recommender systems typically store personal preference profiles. Many items in the profiles can be represented by numerical attributes. However, the initial profile of each user is incomplete and imprecise. One important problem in the development of these systems is how to learn user preferences, and how to automatically adapted update the profiles. To address this issue, this paper presents an unsupervised approach for learning user preferences over numeric attributes by analyzing the interactions between users and recommender systems. When a list of recommendations shown to a target user, the favorite item will be selected by him/her, then the selected item and the over-ranked items will be employed as valuable feedback to learn the user profile. Specifically, two contributions are offered: 1), a learning approach to measure the influence of over-ranked items through analysis of user feedbacks and 2), a weighting algorithm to calculate weights of different attributes by analyzing user selections. These two approaches are integrated into a traditional adaption model for updating user preference profile. Extensive simulations and results show that both approaches are more effective than existing approaches.

1. Introduction

Recommender Systems (RS) [1–7] are widely used mechanism for providing advices to those who want to select preferred choices from a set of items, products or activities. During the decision-making process, however, users are often overloaded with large amount of continuous data. This is especially the case on the Internet, with examples being at which hotel to stay or what news to read in the morning. Users need to distinguish between what they are most interested in and what is less interesting, but it is almost impossible to manually filter out every piece of information and evaluate it accurately. With this in mind, modeling and analyzing user preferences has become one of the methods utilized in RS, which is designed to enable all incoming data to be rated [8–10], ranked, and filtered according to stored user preferences. In order to generate a proper recommendation ranking list, the system should store an appropriate profile of the user. Hence, to some extent, this work fits into the "learning to rank" field of research [11–13].

A user preference profile [14–17], stores personal profiles associated with each user's preference and is one of the most direct solutions for decision-making problem. User numerical preferences are recorded in their file, and it is important for a RS to predict which items would be selected in the future since it is necessary to rank candidate items according to the profile. The user profile must somehow represent the preference of the user with respect to each of the possible values of the attributes. Evaluating which items better fit the user preference profile is of equal importance to determining which items are to be recommended to the real user. However, it has been proven that the initial user preference profile is usually incomplete and imprecise, as discussed in [18], or even empty. That is why it is necessary to use methods that update and modify the initial information about a user profile. In fact, learning and adapting profiles are current widely studied research areas. A user profile can be developed by extracting patterns from training data, e.g., using clustering techniques. In order to automatically acquire and learn user preferences [14,19–21], the RS requires some kind of information from the users to guide the learning process. This feedback may be obtained implicitly, explicitly or by combining both approaches [22].

Explicit feedback is obtained by forcing users to evaluate items, indicating how relevant or interested they are in them by using a numeric scale. Systems with explicit feedback offer high performance and simplicity [23–26]. Obviously, users are usually reluctant to spend time and effort giving explicit feedback. Time-cost and user-passive participation are two major limitations in obtain-

* Corresponding author.
E-mail address: liuyun@bjtu.edu.cn (Y. Liu).

http://dx.doi.org/10.1016/j.knosys.2016.09.003
0950-7051/© 2016 Elsevier B.V. All rights reserved.

ing useful feedback. [27] stated that only about 15% of users supply explicit feedback even if they were encouraged to do so.

In contrast, implicit feedback can be obtained by monitoring interactions between users and the RS, and then automatically capturing user preferences. The amount of data collected is very large, and the computation needed to derive recommendations for adaptations is extensive, Thus, confidence in the suitability of these adaptations is likely to be relatively low. Implicit feedback has been used in many previous methods [28–30], and these existing methods have shown promising results.

For these reasons, we propose an adapted and unsupervised method to update the initial user preference profile, which leverages user interactions and does not require any explicit information from users. It is quite usual that recommender systems have to be used in settings in which the user has to make decisions very often. Thus, we assume that the system is used regularly, such that useful implicit feedback can be continually obtained.

In this paper, we provide a detailed algorithm for the adaptation of the user profile after the analysis of each selection by user. The main goal is to obtain an adequate recommendation ranking list and satisfactory user selection by adapting the proposed preference functions over time. Firstly, there is a set of candidate items that can be recommended to a user, represented by a set of numerical attributes. For a decision problem, these items are evaluated and ranked by the recommender system based on the information stored in the user preference profile. Then the available items will be shown to user in an ordered way (e.g. the system could evaluate and order the latest morning news and present the recommendation list to the user). When the ranking list of candidate items is shown to a user, his (her) favorite item will be selected. Finally, the selected item and over-ranked items (items that were ranked above the selected one), which can be collected as implicit feedback, will be analyzed in order to decide how to adapted update the user profile. On the one hand, obviously, the selected item should be treated as important feedback because it reflects a user's preference to a large extent. On the other hand, the over-ranked items (ranked above the selected item but not selected by the user) are evaluated according to the former user profile. This means that, at this time in the decision problem, the over-ranked items are great recommendations based on the stored user profile, but are not the ideal recommendations for the target user. The over-ranked items affect the user profile in a negative way and the negative influence should not be ignored. Hence, the over-ranked items are also important implicit feedback, which can be used to update the user profile. More specifically, there are two aims for the dynamic adaptation algorithm: to modify the user profile so that items similar to the selected one have a higher ranking (the best case: top one of the recommendation list) in the future, and to modify the user profile so that items similar to the ones that were ranked above the selected item have a lower ranking in future decisions. Although there similar previous works exists [2,9], two important aspects have not been fully considered: the weight of each over-ranked item and the weights of different attributes. In order to solve these two problems, we proposed the following solutions, which offer a two-fold contribution:

- The proposed adaptation method pays more attention to analyzing items that ranked above the selected one and a proper logit function is used to define the influence of different over-ranked items. Each over-ranked item is assigned a weight to make the updated profile more accurate.
- In a real situation, users are usually concerned with one or two significant attributes when selecting the items recommended by a RS. We therefore propose a moving-window-based method to define weights of different attributes.

The rest of the paper is organized as follows: Section 2 includes a brief description of previous related work. Section 3 describes the whole process of recommendation prior to the process of dynamic adaptation update. Section 4 presents a novel model on learning user preference profile, including the traditional adaptation model, the influence of different over-ranked items and calculation of the weights of different attributes. Section 5 describes the evaluation procedure, and provides encouraging results. Finally, Section 6 gives conclusions and outlook for further research in this area.

2. Related work

The main focus of this paper is to present a weighted adaption-updating algorithm that automatically learns user's preferences thru the values from numerical attributes. As stated in [31], there are four main approaches to recommendation: content-based systems, collaborative filtering, demographic systems and knowledge-based systems. This paper presents a knowledge-based system such as the one depicted in Fig. 1. The whole dynamic adaptation framework includes four main parts: using numerical attributes to represent user preference (more details in Section 3.1); generating a recommendation ranking list shown to user (Section 3.2); obtaining useful user feedback (Section 3.1) and adapting the user preference profile (Section 3.2) by analyzing the feedback.

Normally, there are two types of information in user profiles: demographic and domain-dependent preferences. The former include information such as sex age, nationality, and height. This information can be used to measure the similarity between users [32]. This information is usually applied to collaborative-based systems and it is not the focus of our work. In addition, many workflow-based methods [33] are usually applied to collaborative-based systems. The collaborative-based technique will be not used in our case, because it falls outside the scope of this work. In this paper, we apply knowledge-based recommendation mechanisms according to the individual preferences of a single user. For each single user, we consider only the problem of learning the preferences of the single user from the analysis of their individual feedback. Most recommender systems focus on a particular domain of application. Each of these systems collects some specific piece of information about some specific domain, Subsequently, this information can be leveraged to improve knowledge about user preferences. However, these approaches are domain-dependent and not generic. For instance, similar to the method of our proposal, literature [34] sorts the search results by employing the user preference, using a basic assumption very similar to the one of this work. They interpret positive and negative feedback in terms of binary preference relationships, and finally a SVM is used to update the user profile. Just to give another domain-dependent example. [35] utilized an adaptation mechanism to update the user profile by taking into account the features of the selected resource. However, one of the drawbacks of this framework is the need for tagged information attached to each resource. Most of the existing techniques for implicit dynamic adaptation of the user profile are heavily centered on a particular domain of application, whereas the proposed method in this paper is generic and domain-independent. We are only concerned with user interaction and the recommender system and thru this collect relevant implicit user feedback. Thus, our approach is directly applicable to any domain.

In general, two of the most common RS tasks are the representation and the management of user interests through a user profile. The user profile represents preferences about a set of attributes by which the alternatives of the recommendation problem are evaluated [8]. Some previous research has addressed this topic. For instance, in [27] the user profile contains a set of words and associated probabilities in which users are interested. This evidence is

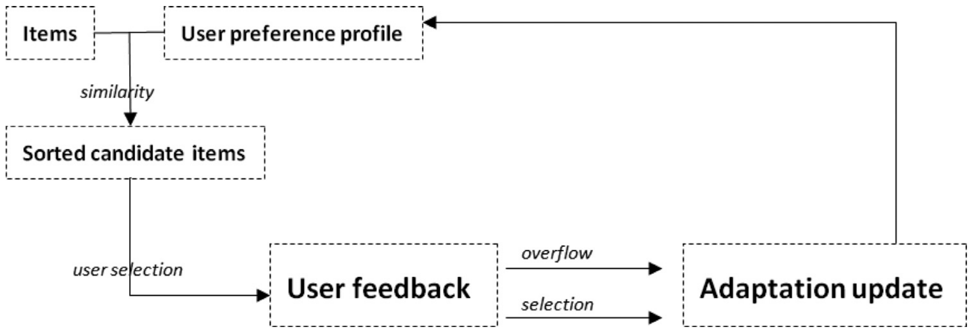


Fig. 1. Adaptation updating system architecture.

then used to rank websites before the recommendation process. In [36], a Bayesian network was used to represent relationships between different types of user-based words, and [25] fuzzy linguistic variables were used to model user preferences. In [18], the authors put forth that a user preference profile can be deduced from a training data set, manually initialized (e.g. via a questionnaire,) and learned by using machine learning techniques, or initialized through a predefined set of stereotypes.

As pointed out in [18], the initial user profile can be empty, or can be obtained with various techniques. There are several ways in which preferences may be represented in the user profile. For example: user profiles can contain a vector of values in which the user is interested [37]; user profiles that contain qualitative preferences represented with fuzzy terms [24]; user profiles containing preferences based on numerical attributes [34]; and user profiles linked with ontology-based semantic information. The information contained in those profiles is used to rank the set of the items to recommend according to user preferences. Note that most of these profiles focus on a single kind of preference values (e.g. recommending a hotel taking into account only the price per night). Hence, it has been recently pointed out by several researchers [32,38] that multi-attribute decision making tools are appropriate when dealing with different attributes at the same time. Many items can be described by different attributes. The main issue of the multi-attribute decision making tools is to find a proper way to combine the user's preferences for all attribute values to determine the overall preference about a given item. For this purpose, a proposal that considers multi-attribute is discussed in [39]. In this case, news are described by the attributes of aboutness, coverage, novelty, reliability and timeliness. The user profile contains information on the preferences of the user in these aspects. The authors define a function by considering each of the selected specific attributes, whereas in our approach all the attributes are considered and modified with the same process. Similar to the approach in [10], each attributes will be considered individually in the work presented in this paper, Unlike [10], the adaptation algorithm of the user profile in this paper will be realized through the analysis of user interaction with the RS. The approach in this paper is also similar to the recent proposal by [5], which use several attributes to express the profile [5]. In that approach, the user profile contains information on the numerical preferences of the user, and some related items, which can be treated as implicit feedback, which are then used to learn the user profile. The authors define all the attributes as of the same weights, whereas in our method different attributes will be modified with different weights.

One of the most well-known problems in RS is the "cold start" problem. The cold start problem [40] occurs when it is not possible to make reliable recommendations due to an initial lack of information. We can distinguish three kinds of cold-start problems: new community, new item and new user. Our proposal fits in considering the last kind of problem, as each profile of a new user is empty. As we discussed before, the initial user profile sometimes can be empty, so the "cold user profile" (empty user profile) can be considered as an exceptional case in learning a user profile. The proposed method will have the ability to deal with the so called "cold start" problem, that is to say that an empty user profile also can also be updated by analyzing user behaviors.

Finally, it can also be argued that this work fits into "learning to rank" research, since the first module of the recommender processes the received items and rates them according to the profile of the user. The information stored in the user profile is used to rank the set of recommendations based on user's preference. Therefore, if the recommender system stores an ideal user preference, then an ideal recommendation ranking list can be shown to the target user. Several researchers have proposed different solutions to accomplish this task, such as thru the use of Machine Learning techniques [41], which allow for combination of several attributes. Showing an ideal recommendation ranking list to user is the final purpose of recommender system. In our proposal, an ideal recommendation ranking list will be generated by learning and generating an ideal user profile. Moreover, the use of a standard index as developed in the "learning to rank" field can be utilized for evaluation of measures.

3. Preliminaries

This section describes the process of recommendation prior to the process of dynamic adaptation update. Firstly, some suitable candidate items must be provided to target user according to his/her user preference profile. Secondly, the most suitable items can be selected by user according to his/her preference. Then the selected item and over-ranked items will be collected as the valuable implicit feedbacks, which can be used to reflect and update the preference profile. Sections 3.1 and 3.2 describe the representation of an initial user preference profile and candidate items. These two sections form the basis for data preparation. Section 3.3 explains how to generate a recommendation ranking list, which will be recommended to a target user.

3.1. Representation of user preference profile

For an appropriate description, user preferences are usually represented or mapped as numerical values. Assuming an item V with n attributes is represented by some numerical attributes, $V = \{v_1, v_2, v_3,\ldots, v_n\}$, where v_i stands for the numerical value of attribute i. If V is selected by a target user, the values will be stored in the user preference profile, which can be used by RS to perform customized recommendations. Therefore, the user preference profile will indicate the user's preference about this type of items. There are two noteworthy aspects of a user preference profile in our approach:

(1) Only numerical values are discussed in this paper.

(2) Although a user preference profile can be generated through many methods, the initial profile is always incomplete.

We define the initial user preference profile as $P = \{P_1, P_2, P_3,\ldots, P_n\}$. Each value of an attribute has its own bounds, in a range as $\Delta r_i = r_i^{max} - r_i^{min}$, where r_i^{max} and r_i^{min} are the maximum and minimum value of attribute i.

3.2. Representation of candidate items

The representation of candidate item is the same as for a user preference profile, which is $C = \{C_1, C_2, C_3,\ldots, C_n\}$, where c_i is the numerical value of attribute i. Candidate items are generated according to the similarity between items and the user preference profile. Before the process of user selection, a list of sorted candidate items must be prepared. We define $L = \{C_1, C_2, C_3,\ldots, C_m\}$ as m candidate items, where C_i stands for an item at mth position.

3.3. Process of recommendation

As discussed previously, candidate items are generated based on the similarity between items and user preference profile. Then, a group of recommendations are sorted according to similarity scores.

The similarity between user preference profile P and an item C is computed through the following formula:

$$sim(P, C) = \frac{1}{n}\sum_{i=1}^{n}\frac{|c_i - p_i|}{\Delta_{r_i}}, \quad (1)$$

In real circumstances, users may not treat all attributes equally. Usually they are concerned with only one or two attributes when selecting candidate items that are recommended to them. For example, a business person will pay close attention to "location" when needing to select from the recommended hotels, while a hotel can possess multiple attributes, such as "price", "location", "service", "stars" .etc. Hence, during the process of recommendation, different weights should be considered. If so, the similarity will be given by:

$$sim(P, C) = \frac{1}{n}\sum_{i=1}^{n}\frac{|c_i - p_i|.w_i}{\Delta_{r_i}}, \quad (2)$$

where w_i is the weight of attribute i. (How to determine different weights of different attributes is discussed in following sections.) Based on the values for similarity, a list of sorted recommendations is shown to the target user. The system assumes that the item ranked at first place most matches the user's preference and the user would choose this one among the group of candidate items as the most preferable. This comes about because the recommendations are generated by evaluating the stored user profile, and the profile will match the user's real preference. However, as discussed before, sometimes the user profile is inaccurate, and the system can not achieve a satisfactory feedback (user does not choose the top item of the recommendation list). Therefore, we need to find an effective method to update the initial user preference profile.

4. Adaptation update

4.1. User feedback

When the sorted recommendations are shown to the target user, the next process is user selection, where we will also obtain relevant user feedbacks. User feedback consist of two components: the selected item and over-ranked items (a set of items ranked above the selected one). The selected item is the most important feedback, which is deemed implicit feedback. As stated above, the selected item directly reflects a user's real preferences. The over-ranked items are also important feedback and should not be neglected.

Table 1
Recommended items and user selection.

		Attribute 1	Attribute 2	Attribute 3	selection
overflow area	Item 1	20	3	45	○
	Item 2	18	3	40	○
	Item 3	16	2	35	○
	Item 4	20	4	39	○
	Item 5	15	3	38	●
	Item 6	40	4	40	○
	Item 7	38	2	50	○

The example shown in Table 1 represents how a user selects an item from the sorted alternatives. As depicted in Table 1, seven items are recommended to a user, and item 5 was selected based on the user preference, item 1–4 were not selected and they are shown in the overflow area. Thus, we have two important types of user feedback from this interaction with the RS:

- User selection (item 5): Obviously, user selection will directly reflect a user's preferences to a certain extent. However, the selected item was not ranked at the top of the recommendation list, which means that we didn't generate a perfect result recommendation. Therefore, the value of each attribute of user profile need to be updated close to the selected item to a certain extent, so that items similar to the selected one have a higher future ranking. Moreover, the values of a user profile are not replaced by the values of a selection directly, because the selected item is also not the user's ideal preferences. Thus, we need a learning method to update the user profile through multiple interactions.
- Overflow area items (the over-ranked item 1–4): These items are ranked in the front part of the recommendation list according to user profile but not selected by user. Thus, these items indirectly reflect user's "negative" preference. That means the user profile should be updated far away from the overflow area, because the overflow area affect the user profile in a negative way. The negative influence should not be ignored. We should to modify the user profile so that items similar to the ones that were ranked above the selected item have a lower ranking in future decisions.

4.2. Coulomb's law and motivation

The Coulomb's Law [2,32] states that "the magnitude of the electrostatic force of interaction between two point charges is directly proportional to the scalar multiplication of the magnitudes of the charges and inversely proportional to the square of the distances between them". This relationship forms the inspiration for our approach, as discussed in [42], it is assumed that there is an 'attraction force' between user preference profile and the selected item, because the selected item reflects to a certain extent a user's preferences. Items that are not selected but ranked among the top few will repulse user preference profile. Finally, the values of user preference profile can be 'attracted' and 'pushed away' toward better values, which are closer to the ideal preferences of the target user.

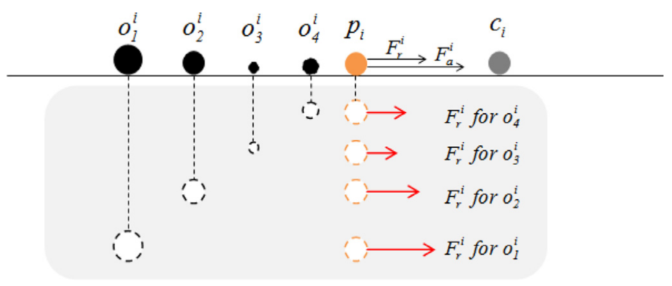


Fig. 2. repulsion forces of different over-ranked items.

4.3. Traditional adaptation process

As stated above [2,8,42], our adaptation was inspired by Coulomb's Law. The values of user preference profile are assumed to be 'attracted' by an 'attraction force' between the user preference profile and the selected item; meanwhile, values will be 'pushed away' by 'repulsion forces' between user preference profile and over-ranked items. Based on this definition, the attraction force of attribute i will be defined as:

$$F_a^i(p_i, c_i, \omega) = \begin{cases} \frac{\Delta r_i.(c_i - p_i)}{|c_i - p_i|^{\omega}} & if\ c_i \neq p_i \\ 0 & if\ c_i = p_i \end{cases} \tag{3}$$

where c_i is the value of attribute i in the chosen alternative, p_i is the value of attribute i stored in the user preference profile, ω is the Parameter. The definition of repulsion force of attribute i is similar to the attraction force, that is:

$$F_r^i(p_i, P^j, \{c_i{}^j\}, \omega, k) = \begin{cases} \sum_{j=1}^{k} \frac{(p_i - c_i{}^j).\Delta r_i}{|p_i - c_i{}^j|^{\omega}} & if\ c_i \neq p_i \\ 0 & if\ c_i = p_i \end{cases} \tag{4}$$

where $c_i{}^j$ stands for the value of attribute i of the jth over-ranked item. Then both the attraction force and repulsion force can be summed up and the final force of attribute i can be defined as:

$$F^i = \theta \times F_a^i + F_r^i \tag{5}$$

where θ is the parameter to control the final force.

This is the traditional model, which has been discussed in [8]. However, two important aspects were neglected:

- Over-ranked items make different contributions in the process of computing repulsion force. (The concern here is about how to improve algorithm performance)
- When users select an item with a few attributes, they only pay attention to one or two attributes. (The concern here is how to obtain agreement with the actual situation)

4.4. Influence of over-ranked items

This section discusses the influence of items in the overflow area, and proposes a logit function to define the different influences of over-ranked items. Each item in the overflow area is analyzed in detail.

Take Table 1 for example, we have four items in the overflow area. Item 3 is very similar to the selected item (item 5). The arrangement reminds us that the item 3 makes the least contribution in computing repulsion force. A mathematical hypothesis is proposed where different items will contribute in varying degrees to computing the repulsion force. That means different over-ranked items should be assigned different proportions in the process of computing the repulsion force. As shown in Fig. 2, for attribute i's value, four over-ranked items o_1^i, o_2^i, o_3^i, o_4^i causes a repulsion force, and the repulsion force contains four repulsion forces caused by over-ranked items. The selected item c^i causes an attraction force. Both forces are applied on attribute i's value for the user preference profile, as discussed in section 4.2.1. In this example, the size of the black ball stands for the similarity between over-ranked item and the selected item and the length of red arrow stands for the probability (strength, proportion) of the repulsion force caused by an over-ranked item. Over-ranked item o_3^i and the selected item are the most similar and item o_3^i 's repulsion force proportion (probability) in terms of the whole repulsion force is the least. Based on the above discussions, logit function [7] is proposed to compute the probabilities.

```
INITIALIZATION
    O^k(o_1,..., o_n),    // over-ranked item k
    V^i(v_1,..., v_m),    // over-ranked items for attribute i
    C(c^1,..., c^n),    // the selected item
    n,    // number of attributes
    m,    // number of over-ranked items
BEGIN
    for each attribute i do
        R^i(r_1,..., r_m) = reranking(V^i(v_1,..., v_m)) // compare with the selected item
    for each over-ranked item j 's attribute i do
        mapping $r_j^i$ = $\hat{r}_j^i$ // mapping each over-ranked item into an intermediate variable
        probability = $P_j^i$ // probability of attribute i of each over-ranked item
    for each over-ranked item j do
        total probability = $P_j$ // probability of each over-ranked item
END
```

Fig. 3. Probability of each over-ranked item.

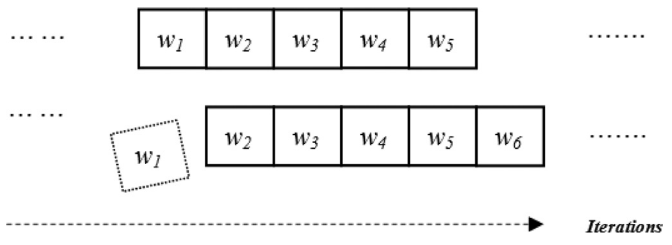


Fig. 4. Example of moving window of attribute i (window size is 5).

Firstly, every attribute i of each over-ranked item is compared with attribute i of the selected item, then we achieve a new ranking vector for every attribute i of each over-ranked item. This is defined as $R^i = \{r_1^i, r_2^i, ..., r_m^i\}$. Take Table 1 for example, the ranking vector of attribute 1 is {3, 2, 1, 3}. Secondly, it is convenient to map each r_j^i into an intermediate variable $\hat{r}_j^i$, ranging in $[-\infty, +\infty]$:

$$\hat{r}_j^i = \frac{1}{2} \ln \left[\frac{1 + 2.(0.5^{(r_j^i + 1)} - 0.5)}{1 - 2.(0.5^{(r_j^i + 1)} - 0.5)} \right] \in [-\infty, +\infty], \tag{6}$$

Thirdly, the probability of attribute i of each over-ranked item j is given by:

$$P_j^i = \frac{\exp(\hat{r}_j^i.r_j^i)}{\sum_{j=1}^{m} \exp(\hat{r}_j^i.r_j^i)} \in [0, 1], \tag{7}$$

With such transformations, the probable value of over-ranked item j's attribute i is achieved. Finally, the probability of each over-ranked item j is computed as:

$$P_j = \frac{\sum_{i=1}^{n} P_j^i}{n}, \tag{8}$$

```
INITIALIZATION
   $C(c^1,..., c^n)$,   // the selected item
   $P(p^1,..., p^n)$,   // user preference profile
   k,   // size of moving window
   $\Delta$,   // range of attribute
   m,   // number of iterations
BEGIN
  size = 0
  for (t=1,t<m,t++) // for each attribute i do
      weight of attribute i = $w_i$
      size++
      if size <= k
         store $w_i$ into moving window orderly
         weight of attribute i = average(each stored $w_i$)
         else if moving size > k
            size--
            delete $w_i$ in first position of moving window
            move forward the moving window
            store $w_i$ into moving window orderly
            weight of attribute i = average(each stored $w_i$)
         end
      end   return (weight of attribute i)
  end
END
```

Fig. 5. Computing weight of each attribute.

where n stands for the total number of attributes. These probabilities are then used to control the process of computing a repulsion force. The pseudo-code in Fig. 3 shows the proposed algorithm.

4.5. Weights of different attributes

As stated in our previous work [9], only one or two attributes become the focus of user attention, which means different attributes will be of different weights. Therefore, we need to evaluate the weights of different attributes to provide a more accurate recommendation.

The set $W = \{w_1, w_2, w_3, ..., w_n\}$ is defined as weights of different attributes. After a process of interaction between the user and the RS, the selected item and the updated values stored in the user

Table 2
Range of five attributes.

Attribute	Range	Unites
Price (v_1)	60–600	RMB
Distance to City Center (v_2)	0–20	km
Distance to Airport (v_3)	0–20	km
Room Size (v_4)	10–60	M^2
Star (v_5)	1–5	⋆

Table 3
Values and weights of three users' ideal preference.

	Traveler		Business man		Student	
	value	weight	value	weight	value	weight
V_1	**150**	**0.2**	**300**	**0.1**	**100**	**0.8**
V_2	**5**	**0.6**	**10**	**0.2**	**15**	**0.05**
V_3	**15**	**0.1**	**10**	**0.4**	**20**	**0.05**
V_4	**25**	**0.05**	**40**	**0.1**	**20**	**0.05**
V_5	**3**	**0.05**	**5**	**0.2**	**2**	**0.05**

preference profile can be treated as important evidence to represent a weight in the next interaction period. Based on our analysis, user preference profile becomes increasingly accurate, and the distance of every attribute between user preference profile and user selection will become increasingly closer. The weight of attribute i is learned by:

$$w_i = \left(1 - \frac{|c_i - p_i|/\Delta_i}{\sum_{i=1}^{n} |c_i - p_i|/\Delta_i}\right)/n - 1, \tag{9}$$

In order to collect enough evidences to measure the weight, we define a moving window (as shown in Fig. 4) to store the values of different attributes in several successive interactions [9].

When the whole window moves to the next interaction, the weight stored in the earliest window is abandoned, and then the second weight becomes a new first one stored in the first position of the moving window. Next, the third one becomes a new second one, and so on. If the moving window size is s, the final weight of attribute i, which will be used in the next interaction, can be summed up by the following formula:

$$W_i = \frac{1}{s} \cdot \sum_{t=1}^{s} w_i^t, \tag{10}$$

The pseudo-code in Fig. 5 shows the proposed algorithm for computing the weight of each attribute.

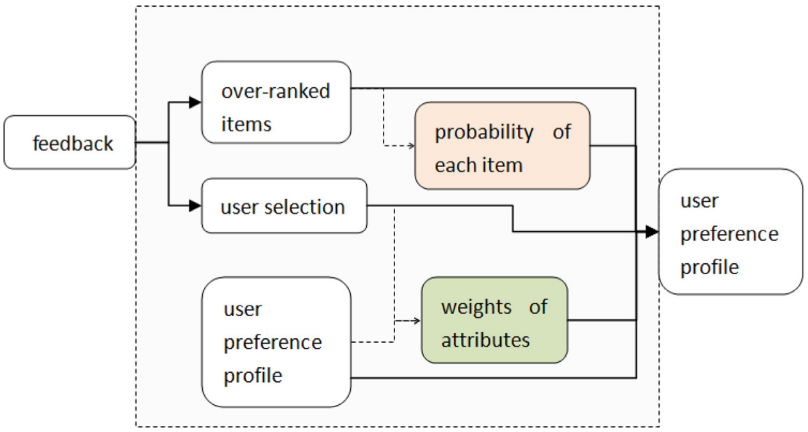


Fig. 6. Learning user preference profile.

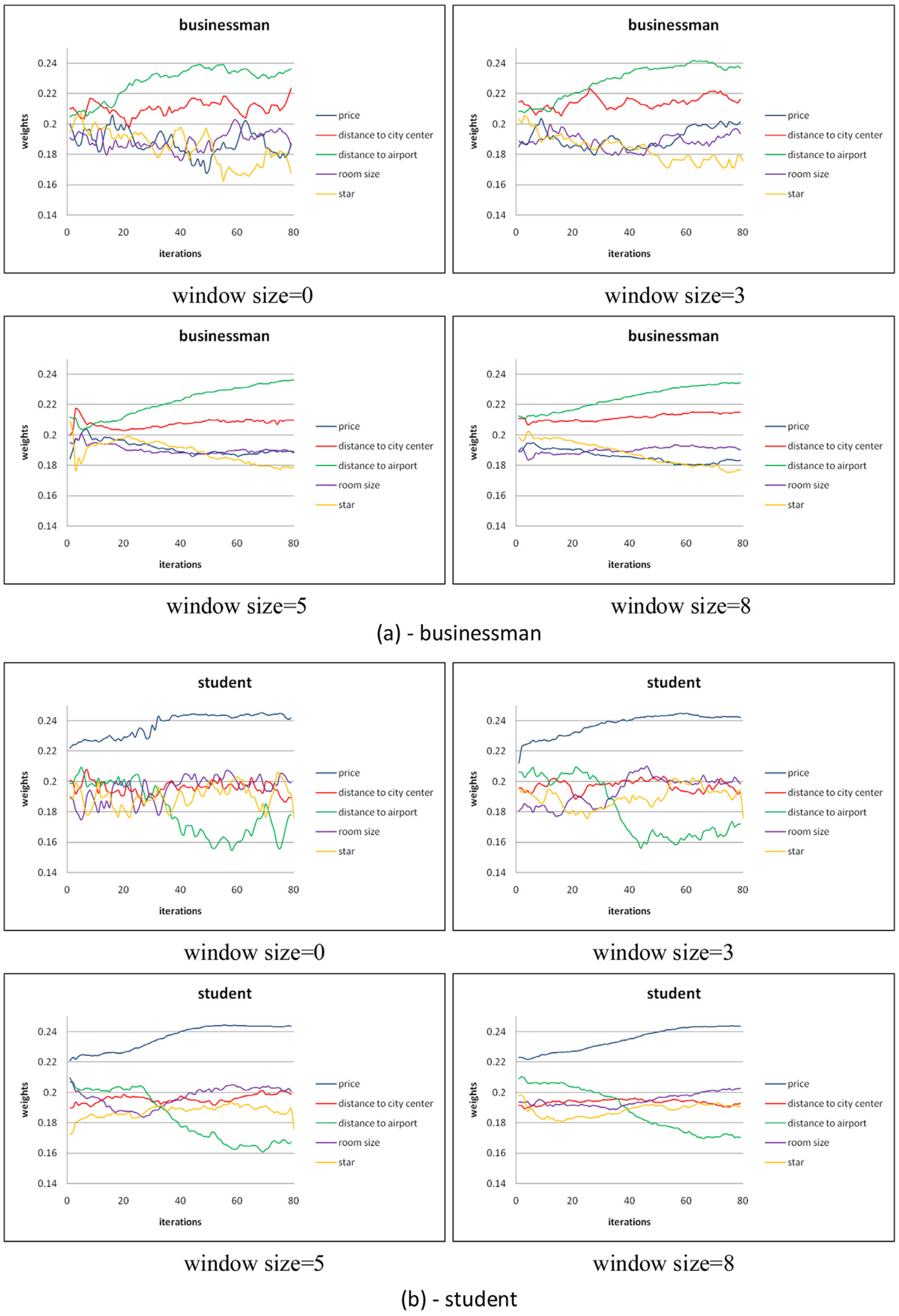


Fig. 7. Evaluation of the weights of different attributes with different window sizes.

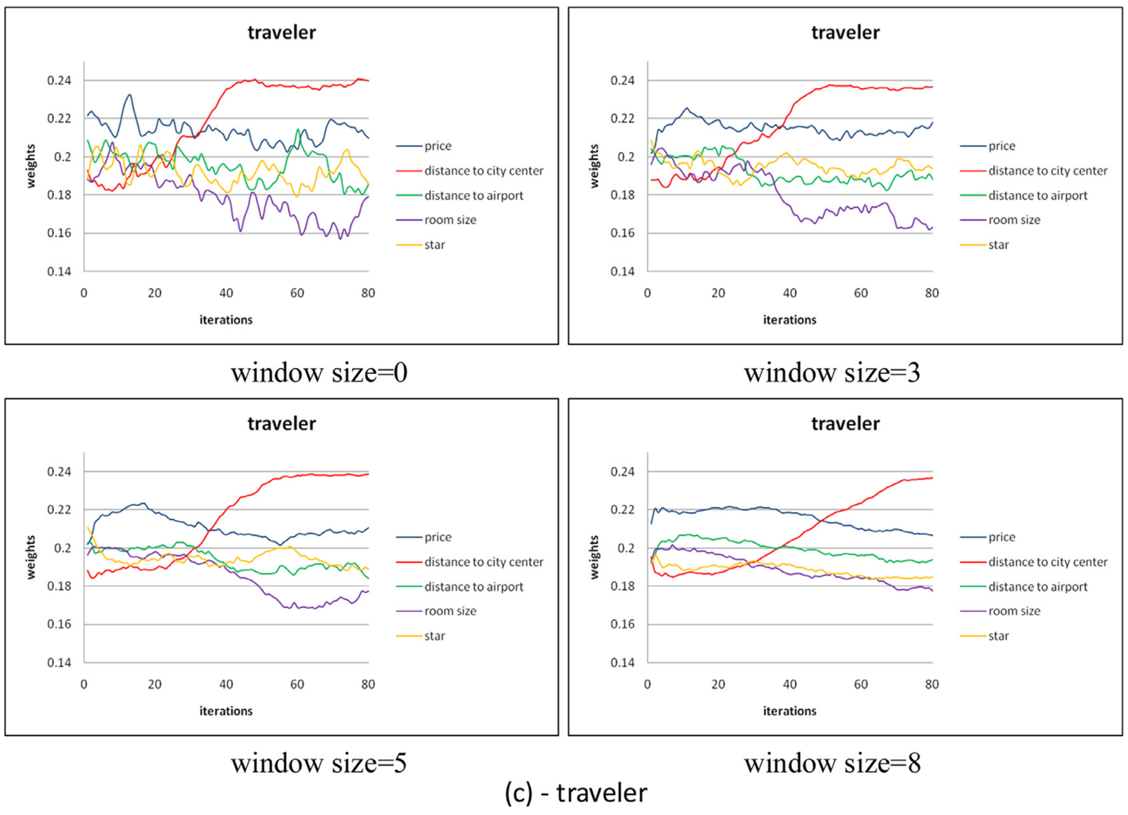


Fig. 7. Continued

4.6. The whole process of learning a user preference profile

So far, all the necessary elements have been collected to complete the model. The attraction force and repulsion force of attribute i in our model can be calculated by:

$$F_a^i(p_i, c_i, w_i, \omega) = \begin{cases} \frac{\Delta r_i.(c_i - p_i)}{|c_i - p_i|^{\omega}}.W_i & if\ c_i \neq p_i \\ 0 & if\ c_i = p_i \end{cases}, \tag{11}$$

$$F_r^i(p_i, P^j, \{c_i{}^j\}, \omega, k) = \begin{cases} \sum_{j=1}^{k} \frac{(p_i - c_i{}^j).P^j.\Delta r_i}{|p_i - c_i{}^j|^{\omega}}.W_i & if\ c_i \neq p_i \\ 0 & if\ c_i = p_i \end{cases}, \tag{12}$$

where P^j is the probability of item j, which was described in section 5.2.2. W_i stands for the weight of attribute i, and it can be achieved by Eq. (6). Then, the final force of attribute i will be calculated by:

$$F^i = \theta \times F_a^i + F_r^i \tag{13}$$

The whole process of learning a user preference profile is depicted in Fig. 6.

5. Experiments and results

In order to test the new model on updating user numerical preference, we use the data of restaurants to implement a RS with the ability to analyze users' preferences from their selections. A description of data is given in the first part of this chapter, and the rest parts provide results of evolutions.

The whole process of simulation consists:

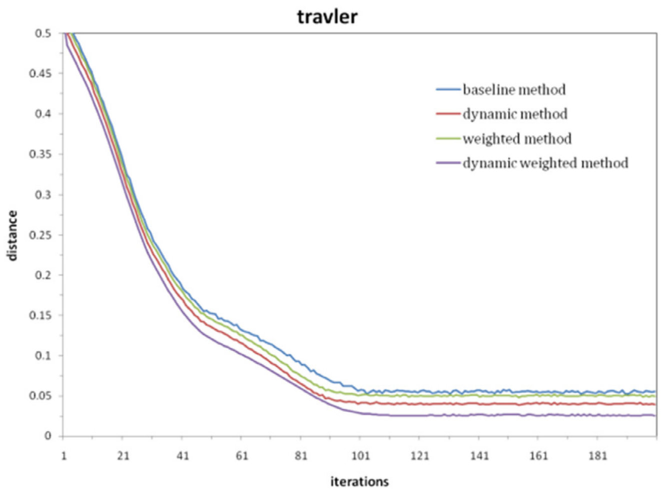
- Ranking 15 candidate items based on current user preference profile. Eq. (2) will be used in the step.
- Simulating the selection of the target user by selecting an item that fits better with his (her) ideal preference.
- Collecting user feedbacks (over-ranked items and the selected item).
- Calculating the influence of each over-ranked item. Eq. (6)-(8) will used in this step.
- Calculating the weights of different attributes. Eq. (9)-(10) will be used in this step.
- Updating user preference profile using Eqs. (11)-(13).

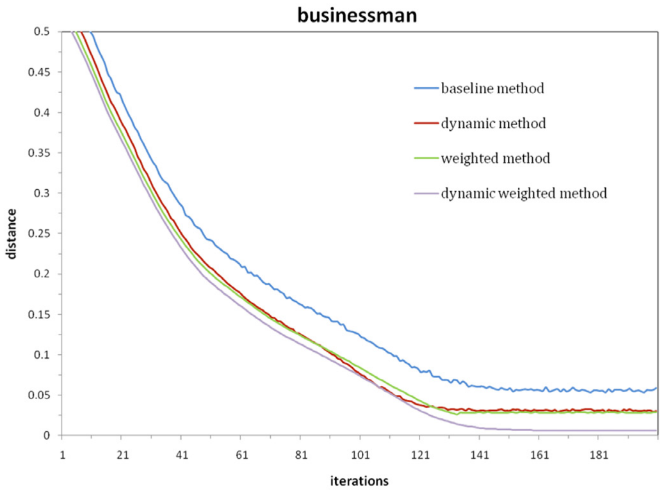
5.1. Data sets

The data sets of this problem were generated according to the "OpenAPI" of Ctrip (www.crip.com, booking hotels) and explanations in [8]. Based on the description of hotels, the hotels datasets were evaluated by five numerical attributes: "price", "distance to city center", "distance to airport", "room size and star". Table 2 shows the range for the five attributes. A group of 1500 hotels with five numerical attributes were generated randomly with a uniform distribution based on the range. Fifteen hotels are ranked independently, providing a total of 100 different recommendations.

The ideal numerical preferences of three types of user (traveler, businessman, and student) were manually defined and three initial profiles were randomly created. Different types of people have different preferences. The values and weights of three users' ideal preference are represented in Table 3.

Take the student group as an example. A student prefers attribute v_1 (price) when selecting an item through the RS, we therefore define the weight of attribute one as 0.8 and the others are defined as 0.05 (total is 1) in order to simulate different weights in the mind of a real user. As for the businessman, comprehensive experience is his major concern. For a traveler, he prefers traveling convenience. In other words different types of users have their unique preferences. The initial weight of each attribute is set to





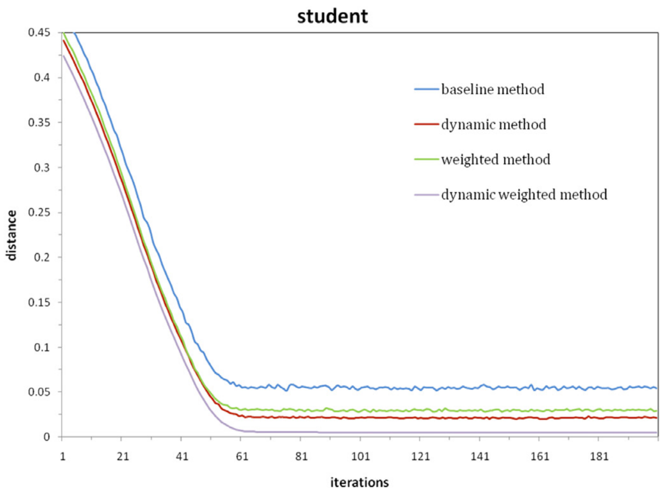


Fig. 8. Average distance between the current and the ideal profile.

0.2, which means that at the beginning of the process of recommendation, each attribute has the same weight value because initially the most important attribute is unknown. The final goal of the adaptation model is to: (1) move the values of a user preference profile closer to the user's ideal preferences; (2) find the most important attribute(s).

5.2. Comparison

In the simulations, the compared methods includes:

(1) **Baseline method:** this method updates the profile by using the model described in section 4.2.2. It was clearly discussed in [2].
(2) **Weighted method:** this is a weighted model to update the profile, which was clearly described in our previous paper [9]. How to calculate the weights of different attributes is discussed in section 4.2.4.
(3) **Dynamic method:** this method updates the profile dynamically by considering the influence of over-ranked items. How to calculate the probability of attribute of each over-ranked item is discussed in section 4.2.3.
(4) **Dynamic weighted method:** this is the proposed method in this paper. This weighted method updates the profile by considering the influence of overflow area. The model is described in section 4.2.5.

5.3. Results

Previous sections have presented a useful model aiming at updating user preference profile by combing the significance of different over-ranked items and weights of different attributes. This section describes the results obtained from experiments. All results shown in this section are the results averages by fifty tests. 200 iterations are used in the whole learning update process.

5.3.1. Evaluation of the weights of different attributes with different moving window sizes

Fig. 7 shows the changes of weights of different attributes with various window sizes. From the evaluations results, we arrived at two important conclusions:

- The most important attribute can be found immediately after several interactions. Take a student for example (Fig. 7(b)), where his most interested attribute is price. In this scenario, the weight value of attribute price rises from 0.2 to 0.25, while the other attributes decline. Meanwhile, the attributes of most concern to a businessman (distance to airport) and traveler (distance to city center) are discovered in a similar mechanism.
- Four window sizes (0, 3, 5, and 8) are evaluated in this paper. The changes of weights become more smooth and steady as the sizes of the moving windows increase.

5.3.2. Evaluation the result of updating user preference profile

In order to evaluate the results of the new learning model, a distance function was used to calculate how different the user preference profile to be updated is for an ideal profile, which represents the exact preferences of the target user. The distance between two profiles was computed by:

$$d(p, c, \Delta) = \frac{1}{n}\sum_{i=1}^{n} |p_i - c_i| / \Delta_i \tag{14}$$

During the simulation process, the distance between a user preference profile and ideal profile was calculated in each iteration.

Fig. 8 compares our method to the other three methods for three types of users and the results show that our method improved the results over previous methods. There was no obvious difference between the results of weighted method and dynamic method for a businessman. A possible reason is that when a busi-

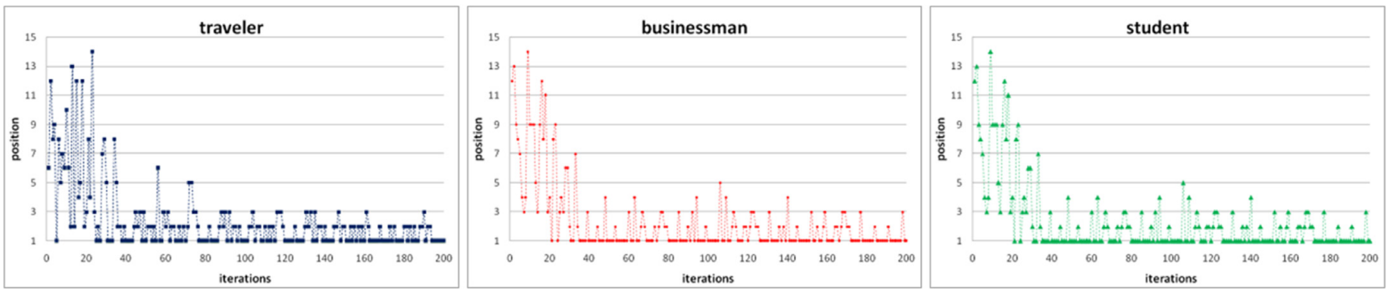


Fig. 9. Ranking position of user's favorite item of recommendations.

nessman selects a hotel, he prefer considering every weight of attributes simultaneously and thus,—different weights for attributes don't influence his choice. In comparison, for students, travelers, the results with the weighted method were better than those of the dynamic method. This reflects a preference for considering s particular attribute (student:-price, traveler:-distance to city center) when choosing a hotel, and this has a larger influence on choice. For example, student preference is prone to lower and more economical prices. They tend to gravitate toward the attribution of price more frequently. The results and analysis above were consistent with characteristics of the ideal profile of the three types of people in real circumstances. Moreover, the results were better than the other three methods after simultaneously considering the weight factors and the influence of the over-ranked items (the proposed model in this paper).

If the initial profile is updated close to the ideal profile, user's favorite item could be ranked in the front positions (such as position 1,2,3 ...) of the recommendation list. That means the recommended ranking list becomes suitable for user preference. Fig. 9 shows the position of the ranking performed by the RS where the user selection is located. The data reflects that the most suitable items can be ranked in the front positions and a user's selection would come out from the front positions after about 30 interactions. And in most cases, user selection was ranked in the first position.

Fig. 10 shows the results of a comparison between the baseline methods and proposed method. The Y-axis stands for the user selection's position (the item selected by a user and its recommended position) among the 15 recommendations. The table behind it selects the statistical number of each user selection's position during 200 iterations. For the front recommended items (position 1,2,3), our method collected more data than the other methods, which reflects an improvement in the performance for updating the profile.

Additionally, as the profile is updated accurately, it helps to substantially improve the accuracy of the ranking of the recommendation list which is based on the profile for offering items to the users. In other words, it is correctly records the user's preference profile as the recommended ranking list is accepted by the user. Therefore, to some extent, this problem can be seen as a problem of learning to rank. An appropriate list of recommendations will be more likely to be accepted by the target user. We adopted the Normalized Discounted Cumulative Gain (nDCG) measurement as a metric, which analyzes a ranking list in a more detailed manner to study the order of the candidates in each step of the learning process. The premise of nDCG is that highly relevant objects appearing in lower positions of a recommendation list should be penalized as the graded relevance value is reduced logarithmically with respect to the position of the object [2]. The equation of the nDGC is

$$nDCG_k = \frac{DCG_k}{DCG_{k-ideal}} \tag{15}$$

DCG$_{k\text{-}ideal}$ (ideal DCG) is the best DCG that we could obtain. DCG$_{k\text{-}ideal}$ is obtained by analyzing the ideal ranking list, which will match the ideal user preference. DCG is obtained by analyzing the recommended ranking list. DCG is defined as:

$$DCG_k = \sum_{i=1}^{k} \frac{(2^{s(i)} - 1)}{\log_2(1+i)} \tag{16}$$

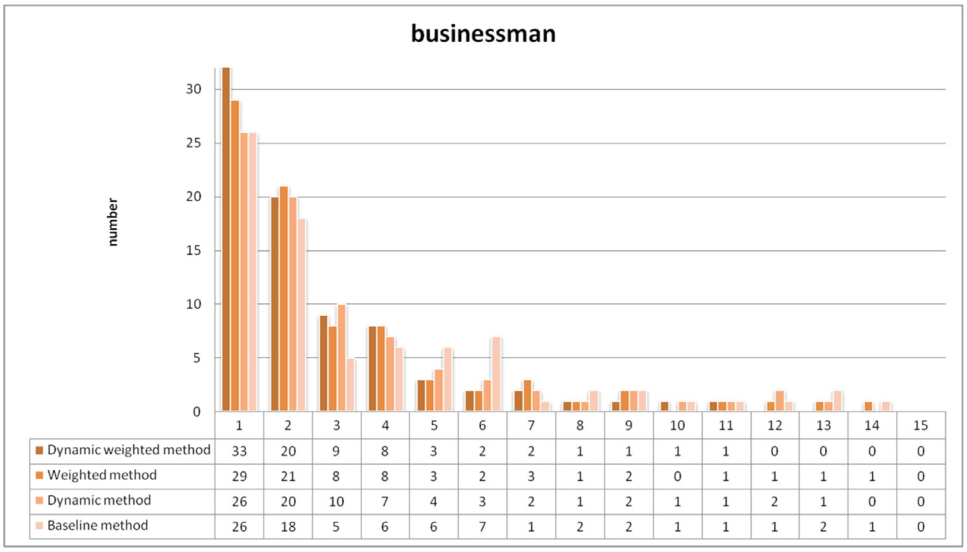
where $s(i)$ is the ranking score at position *i, k* is the total number of the ranking list. In this paper, we focus on the top five candidates, with the candidate at the first position of ideal ranking list getting the highest score 5, and then 4, 3, 2, 1, 0, 0, 0 .... Therefore, a value of 1 in nDCG means that the ranking list of the candidates made by the RS coincides with the ideal ranking.

Fig. 11 shows the evolution of the nCDG$_{15.}$ As the updated user preference profile and the ideal profile get closer, nCDG$_{15}$ increases.

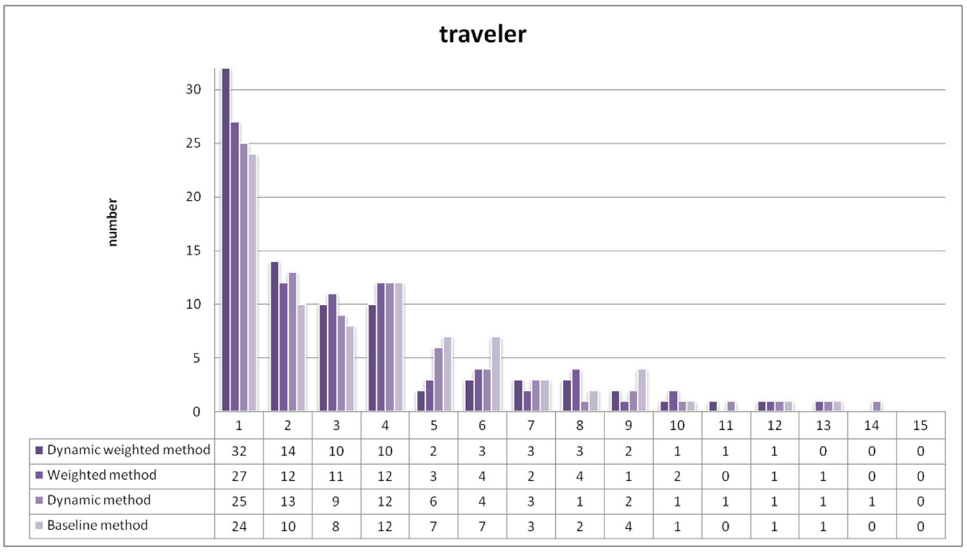
One conclusion from the experimental results with nDCG is that the dynamic method is almost the same as that of weighted method fo the businessman. The results are both superior to the baseline methods. For a student and traveler, the nDCG results from the weighted method is worse than that of the dynamic method. The reason is the same as the results from the analysis of Fig. 7. The results and analysis above were consistent to the characteristics of the ideal profile of the three types of people that this paper has defined. Additionally, the nDCG results were better when it was not computed according to the other three methods after simultaneously considering the weighted factors and the over-ranked items (the model proposed by this paper). For businessmen, according to the model proposed, nDCG is approximately 0.85 and the baseline method was only about 0.68. For travelers, the nDCG is approximately 0.87 and the baseline was only 0.7. For students, the nDCG is approximately 0.78 and the baseline was only 0.59.

5.3.3. Evaluation of the 'cold start' problem

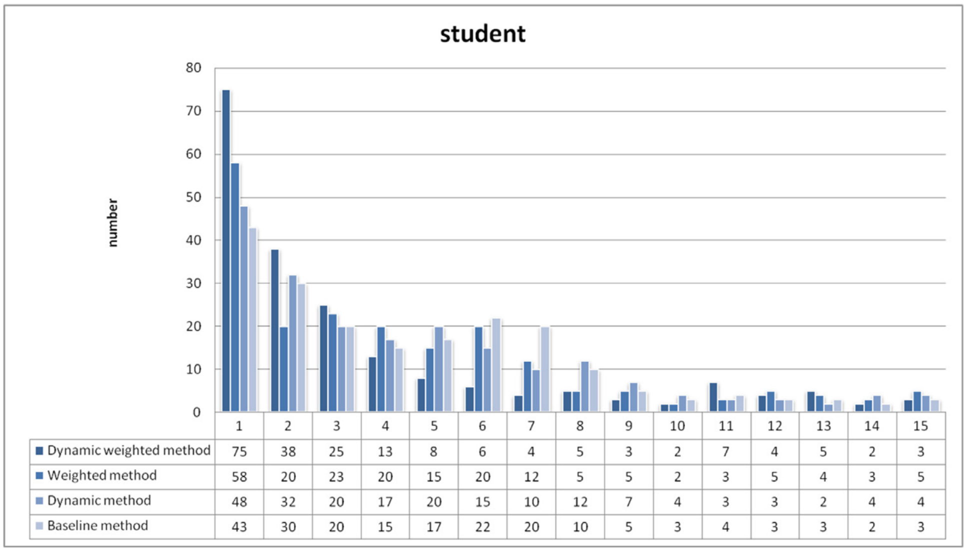
One claim about benefit of our method is the ability to deal with the "cold start" problem, i.e., the recommendation to a user who uses the system for the first time, and as such, does not have any record in the profile before. As we discussed previously, the initial user profile sometimes can be empty, so the "cold user profile" (empty user profile) can be considered as an exceptional case for learning a user profile. To evaluate our approach, we assume a new student, whose initial profile was set to empty values, and was about to use the system to book a hotel. Along with the system that is used regularly, the user profile will be updated to the ideal values by using the proposed method. We used distance function Eq.(14) to evaluate the updated result of each attribute. Fig. 12 presents the distance value between the stored user profile and



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
■ Dynamic weighted method	33	20	9	8	3	2	2	1	1	1	1	0	0	0	0
■ Weighted method	29	21	8	8	3	2	3	1	2	0	1	1	1	1	0
■ Dynamic method	26	20	10	7	4	3	2	1	2	1	1	2	1	0	0
■ Baseline method	26	18	5	6	6	7	1	2	2	1	1	1	2	1	0



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
■ Dynamic weighted method	32	14	10	10	2	3	3	3	2	1	1	1	0	0	0
■ Weighted method	27	12	11	12	3	4	2	4	1	2	0	1	1	0	0
■ Dynamic method	25	13	9	12	6	4	3	1	2	1	1	1	1	1	0
■ Baseline method	24	10	8	12	7	7	3	2	4	1	0	1	1	0	0



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
■ Dynamic weighted method	75	38	25	13	8	6	4	5	3	2	7	4	5	2	3
■ Weighted method	58	20	23	20	15	20	12	5	5	2	3	5	4	3	5
■ Dynamic method	48	32	20	17	20	15	10	12	7	4	3	3	2	4	4
■ Baseline method	43	30	20	15	17	22	20	10	5	3	4	3	3	2	3

Fig. 10. Number of user selection's ranking position (total 200 iterations), e.g. 75 means 75 ranking position 1 was selected by user in the total 200 iterations.

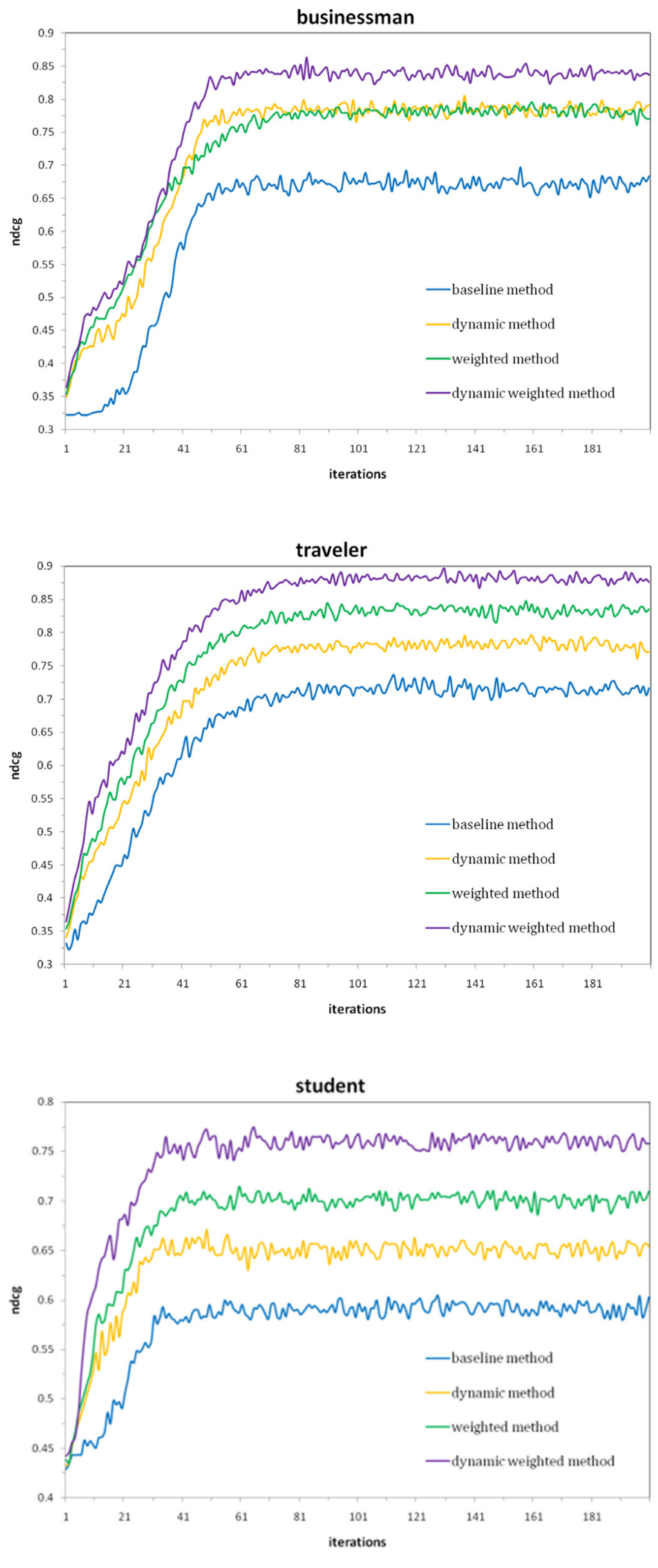


Fig. 11. NDCG analysis performance.

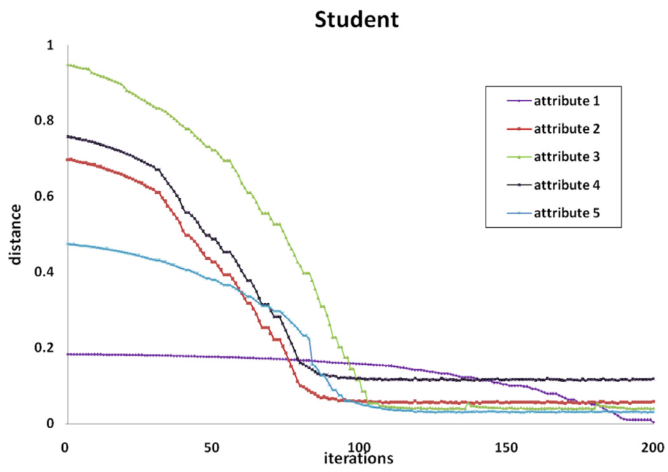


Fig. 12. Evaluation of the "cold student" problem.

ideal profile in terms of all attributes in "cold student" test. The testing showed that all the attributes were updated close to the ideal preference values even if we have an empty initial user profile. In summary, our proposal can handle the "cold start" problem.

6. Conclusion and future work

One of the most challenging tasks in the development of recommender systems is the design of techniques that can infer the preferences of users through observation of their actions. A user preference profile stores user numerical preference information, yet the initial user preference profile is not the most accurate profile and it is necessary to update this profile through analyzing interactions between a user and the RS. Two main contributions have been presented in this paper: 1), a method to compute the probability of each over-ranked item by analyzing the influence of each over-ranked item; and 2), a method to evaluate weights of different attributes.

We note that in real situation, users always pay attention to one or two attributes; therefore, we proposed a method to find the attribute(s) of most interest. The simulation results of showed that our method was able to enhance the performance for learning a user profile, and the weighted algorithm helps in finding attributes of most interest.

The authors suggest that the following four concerns are suitable targets for further study:

- Semantic attributes (comments) and categorical attributes (very big, big, small, very small) could be added to the proposed model. That means a more appropriate method should be created to represent user preferences. User preference information should be widely covered (not just numerical attributes).
- The model should be tested in a practical application platform so that more useful data can be obtained for future study.
- Combining the proposed model with a more efficient recommendation algorithm would be beneficial. An excellent recommendation algorithm is the basis for acquiring user implicit information.
- In different situations, numerical attributes are an interval of numbers rather than just a fixed number. Therefore, a more appropriate method needs to be proposed to represent user numerical preferences.

Acknowledgement

This work has been supported by the Fundamental Research Funds for the Central Universities W16JB00100 and the National Natural Science Foundation of China under Grant 61172072, 61401015, 61271308.

References

[1] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Syst. Appl. 41 (4) (2014) 2065–2073.
[2] L. Marin, A. Moreno, D. Isern, Automatic preference learning on numeric and multi-valued categorical attributes, Knowl. Based Syst. 56 (2014) 201–215.
[3] C.E. Briguez, et al., Argument-based mixed recommenders and their application to movie suggestion, Expert Syst. Appl. 41 (14) (2014) 6467–6482.
[4] A. Abbas, L.M. Zhang, S.U. Khan, A survey on context-aware recommender systems based on computational intelligence techniques, Computing 97 (7) (2015) 667–690.
[5] G. Adomavicius, D. Jannach, Preface to the special issue on context-aware recommender systems, User Model. User-Adapted Interact. 24 (1-2) (2014) 1–5.
[6] G. Adomavicius, et al., Context-aware recommender systems, Ai Magazine 32 (3) (2011) 67–80.
[7] F.E. Walter, S. Battiston, F. Schweitzer, A model of a trust-based recommendation system on a social network, Auton. Agents Multi-Agent Syst. 16 (1) (2008) 57–74.
[8] L. Marin, A. Moreno, D. Isern, Automatic learning of preferences in numeric criteria, Artif. Intell. Res. Dev. 232 (2011) 120–129.
[9] Z. Zhang, Y. Liu, A.Z. Qing, A weighted method to update network user preference profile dynamically, Int. J. Interdiscip. Telecommun. Netw. 2 (6) (2014) 52–67.
[10] G. Adoinavicius, Y.O. Kwon, New recommendation techniques for multicriteria rating systems, Ieee Intell. Syst. 22 (3) (2007) 48–55.
[11] P. Li, C.J.C. Burges, Q. Wu, McRank, *learning to rank using multiple classification and gradient boosting*, in: *Proc. of Twenty-First Annual Conference on Neural Information Processing Systems, NIPS 2007*, British Columbia, Canada, 2007.
[12] N. Ifada, R. Nayak, Do-Rank: DCG optimization for learning-to-rank in tag-based item recommendation systems, Adv. Knowl. Discovery Data Min. Part Ii 9078 (2015) 510–521.
[13] J. Jin, et al., Learn to rank: Automatic helpfulness analysis of online product reviews, in: Proceedings of the Asme International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Detc 2010, Vol 3, 2010, pp. 467–476.
[14] H. Banati, M. Bajaj, Dynamic user profile generation by mining user opinion, in: Third International Conference on Computer Engineering and Technology (Iccet 2011), 2011, pp. 557–562.
[15] L. Chen, G.L. Chen, F. Wang, Recommender systems based on user reviews: The state of the art, User Model. User-Adapt. Interact. 25 (2) (2015) 99–154.
[16] L. Rokach, S. Kisilevich, Initial profile generation in recommender systems using pairwise comparison, Ieee Trans. Syst. Man Cybern. Part C-Appl. Rev. 42 (6) (2012) 1854–1859.
[17] L. Marin, et al., On-line dynamic adaptation of fuzzy preferences, Inf. Sci. 220 (2013) 5–21.
[18] M. Montaner, B. López, J.L. de La Rosa, A taxonomy of recommender agents on the internet, Artif. Intell. Rev. 19 (4) (2003) 285–330.
[19] L. Marin, D. Isern, A. Moreno, A generic user profile adaptation framework, in: 13th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2010, Tarragona, Spain, 2010, pp. 143–152.
[20] T. Chen, et al., Content recommendation system based on private dynamic user profile, in: Proceedings of 2007 International Conference on Machine Learning and Cybernetics, Vols 1-7, 2007, pp. 2112–2118.
[21] C. Haruechaiyasak, et al., A dynamic framework for maintaining customer profiles in e-commerce recommender systems, in: 2005 Ieee International Conference on E-Technology, E-Commerce and E-Service, Proceedings, 2005, pp. 768–771.
[22] I.P. Schwab, W. Koychev, I, Learning to recommend from positive evidence, in: Proceedings of Intelligent User Interfaces 2000, ACM Press, 2000, pp. 241–247.
[23] E. Peis, E. Herrera-Viedma, J.M. Morales-del-Castillo, A semantic service model of selective dissemination of information (SDI) for digital libraries, Profesional De La Informacion 17 (5) (2008) 519–525.
[24] J.M. Morales-Del-Castillo, et al., A semantic model of selective dissemination of information for digital libraries, Inf. Technol. Lib. 28 (1) (2009) 21–30.
[25] J.M. Morales-del-Castillo, et al., Recommending biomedical resources a fuzzy linguistic approach based on semantic web, Int. J. Intell. Syst. 25 (12) (2010) 1143–1157.
[26] L. Sebastia, et al., e-Tourism: A tourist recommendation and planning application, in: 20th Ieee International Conference on Tools with Artificial Intelligence, Vol 2, Proceedings, 2008, pp. 89–96.
[27] M. Pazzani, D. Billsus, Learning and Revising User Profiles: The Identification of Interesting Web Sites, Mach. Learning 27 (3) (1997) 313–331.
[28] L. Baltrunas, X. Amatriain, Towards time-dependant recommendation based on implicit feedback, in: Proc. of Workshop on Context-aware Recommender Systems in conjunction with the 3rd ACM Conference on Recommender Systems, RecSys 09, New York, USA, 2009, pp. 25–30.
[29] E. Cheng, F. Jing, M. Li, W. Ma, H. Jin, Using implicit relevance feedback to advance web image search, in: Proc. of IEEE International Conference on Multimedia and Expo, ICME 2006, IEEE Computer Society, Toronto, ON, Canada, 2006, pp. 1773–1776.
[30] A. Veilumuthu, P. Ramachandran, Discovering implicit feedbacks from search engine log files, Discovery Sci. Proc. 4755 (2007) 231–242.
[31] R. Burke, A. Felfernig, M.H. Goker, Recommender systems: An overview, Ai Magazine 32 (3) (2011) 13–18.
[32] A. Moreno, A. Valls, D. Isern, L. Marin, J. Borràs, SigTur/E-destination: Ontologybased personalized recommendation of tourism and leisure activities, Eng. Appl. Artif. Intell. 1 (26) (2013) 633–651.
[33] L. Zhen, G.Q. Huang, Z.H. Jiang, Recommender system based on workflow, Decis. Support Syst. 48 (1) (2009) 237–245.
[34] T Joachims, F. Radlinski, Search engines that learn from implicit feedback, Comput. Commun. 40 (2007) 34–40.
[35] G. Castellano, et al., Learning fuzzy user profiles for resource recommendation, Int. J. Uncertainty Fuzziness Knowl. Based Syst. 18 (4) (2010) 389–410.
[36] K.-Y. Jung, K.-W. Rim, J.-H. Lee, Automatic preference mining through learning user profile with extracted information, in: Structural, Syntactic, and Statistical Pattern Recognition Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, 2004, pp. 815–823.
[37] O. Phelan, K. McCarthy, M. Bennett, B. Smyth, On using the real-time web for news recommendation & discovery, in: Proc. of 20th International Conference Companion on World Wide Web, ACM: Hyderabab, India, 2011.
[38] A. Valls, M. Batet, E.M. Lopez, Using expert's rules as background knowledge in the ClusDM methodology, Eur. J. Oper. Res. 195 (3) (2009) 864–875.
[39] J.J.P. Arias, et al., Making the most of TV on the move: My newschannel, Inf. Sci. 181 (4) (2011) 855–868.
[40] G. Adomavicius, A. Tuzhilin, 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) (2005) 734–749.
[41] Y. Freund, et al., An efficient boosting algorithm for combining preferences, J. Mach. Learn. Res. 4 (6) (2004) 933–969.
[42] L. Marin, D. Isern, A. Moreno, Dynamic adaptation of numerical attributes in a user profile, Appl. Intell. 39 (2) (2013) 421–437.