A Dynamic Weighted Method on Learning User Preference Profile

Zhiyuan Zhang^{1,2}, Yun Liu^{1,2,*}, Guandong Xu^{3}, Guixun Luo^{1,2}

1, School of Communication and Information Engineering, Beijing Jiaotong University, Beijing 100044, China.

2, Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, Beijing 100044, China.

3, Advanced Analytics Institute, University of Technology Sydney, Australia.

* Corresponding author. e-mail: liuyun@bjtu.edu.cn.

Abstract

Recommender Systems usually 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 the user preferences, and how to update the profiles dynamically. This paper presents an approach to learn user preferences over numeric attributes by analyzing the interactions between users and RS. More specifically, there are two contributions in this paper: 1) a learning approach to measure the influence of over-ranked items through analysis of user feedbacks; 2) a weighing algorithm to calculate weight of different attributes by analyzing user selections. These two approaches are integrated into a traditional model for updating user preference profiles dynamically. Extensive simulations and results show that both approaches are more effective than existing approaches.

Keywords: Recommender System, Numerical attributes, User preference profile, Dynamic

1. Introduction

Recommender Systems (RS) [1-7] is a widely used mechanism for providing advices to those who want to select the preferred from a set of items, products or activities. Users, however, are often overloaded with large amount of continuous data during the decision-making process, especially on the Internet, such as which hotel to stay or what news to read in the morning. Users need to distinguish what they are most interested in and what should be less interested, but it is impossible to manually filter out every piece of information and evaluate it accurately. Under such circumstances, storing and analyzing user preferences has become one kind of methods in RS, which is designed to enable all incoming data to be rated [8-10], ranked, and filtered according to the stored user preference. To some extent, this work is suitable for the "learning to rank" field of work [11-13].

User preference profile [14-17], which stores personal profiles associated to each user's preference, is one of the direct solutions of the decision-making problem. User's numerical preferences are recorded in his/her file, and it is important for RS to predict which items would be selected in the future since it is necessary to rank candidate items according to the profile. User preference profile will reflect target user's preference in a certain extent. Evaluating which items better fit the user preference profile is of equal importance to determining which items are to be recommended the real user. However, the initial user preference profile is usually imprecise, resulting in the difficulty of reflecting the real user's preference. Therefore, dynamic profile updating becomes crucial in preference-based RS.

The representation and automatic reflection of the dynamic evolution of the user interests through a proper user profile are two of the most challenging tasks in measuring and predicting the most suitable items for a target user[10]. Some previous researches have addressed this topic. For instance, in [18] the user profile contains a set of words and associated probabilities in which users are interested. These evidences are used to rank websites before the recommendation process. In [19], a Bayesian network is used to represent relationships between different types of user-based words, and [20] uses fuzzy linguistic variables to model user preferences. In [21], the authors told us that user preference profile can be deduced from a training data set, manually initialized (e.g. via a questionnaire) and learnt by using machine learning techniques, or initialized through a predefined set of stereotypes. However, it has been proved that the initial user preference profile is usually incomplete and imprecise, as discussed in [21], or even empty. That is why we need to focus on the task of the management of the user preferences. In this paper, attention is paid to how to update the user preference profile dynamically by analyzing the interaction between user and RS. In order to fulfill this [14, 22-24], the system must receive relevant useful explicit and implicit feedbacks [25].

Explicit feedbacks are obtained by forcing users to evaluate items, indicating how relevant or interested they are to them using the numeric scale. Systems with explicit feedbacks offer high performance and simplicity [20, 26-28]. Obviously, users are usually reluctant to spend time and effort giving explicit feedbacks. Time-cost and user-passive participation are two major limitations in obtaining useful feedbacks. [18] has proved that only 15% of users would supply explicit feedbacks even if they were encouraged to do so. In contrast, implicit feedbacks can be obtained by monitoring interactions between users and RS, and automatically capturing user preferences. Implicit feedbacks have been used in many previous methods [29-31], and these existing methods have shown promising results. This paper proposes a dynamic and unsupervised method to update initial user preference profiles on numerical attributes, which leverages user interactions and does not require any explicit information from users.

The whole process of building user preference profile consists of two parts: the process of user selection and the process of recommendation . We mainly address the process of user selection, by analyzing the user selections (feedbacks), in which we can learn the user preference and update the profile. Firstly, there is a set of candidate items that can be recommended to user, represented with a set of numerical attributes. Then these items are evaluated and ranked based on the stored user preference profile. When the list of candidates is shown to a user, his (her) favorite item will be selected. Finally, the selected items and over-ranked items (items that were ranked above the selected one) will be analyzed in order to decide which changes must be made to the user preference profile [2, 9]. Although there exist previous works, two important aspects are not fully considered: the influence of each over-ranked item and the influence of different attributes. In order to solve these two problems, we proposed the corresponding solutions in the paper. The contributions of this paper are two-fold:

- The proposed adaptation method analyzes 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 probability (weight) to make the updated profile more accurate.
- In real situation, users usually concern one or two significant attributes when selecting the items recommended by RS. We therefore propose a moving-window-based method to evaluate weights of different attributes.

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

2. 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 becomes the feedback, which can be used to reflect and update the preference profile. Section 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 sorted candidate items, which will be recommended to a target user.

2.1. Representation of user preference profile

For appropriate description, user preference are usually represented or mapped as numerical values. Assuming an item V with n attributes is represented by some numerical attributes, $V=\{v_1, v_2, v_3, ..., v_n\}$, 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 user preference profile:

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

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

We define the initial user preference profile as $P=\{p_1, p_2, p_3, ..., p_n\}$. Each value of an attribute has its own bounds, in a range as $\Delta r = r_i^{max} - r_i^{min}$, where r_i^{max} and r_i^{min} are the maximum and minimum value of attribute *i*.

Commented [GX1]: Δ_r=

2.2. Representation of candidate items

The representation of candidate item is the same as user preference profile, which is $C=\{c_1, c_2, c_3, ..., c_n\}$, where c_i is the numerical value of attribute *i*. Candidate items are generated according to the similarity between items and user preference profile. Before the process of user selection, a list of sorted candidate items must be prepared. We define $L=\{C_1, C_2, C_3, ..., C_m\}$ as *m* candidate items, where C_i stands for an item at mth position.

2.3. Process of recommendation

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

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

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

In real circumstances, users may not treat all attributes equally. Usually they concern only one or two attributes when selecting candidate items recommended to them. For example, a business man will pay close attention to the "location" when he needs to select 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:

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

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 of similarity, a list of sorted candidate items is shown to the target user. The item ranked in the first place matches user's preference most and the user would choose this one among the group of candidate items as the most preferable. This is the whole process of recommendation prior to the process of dynamic adaptation update.

3. Dynamic adaptation update

The main focus of this paper is to present a dynamic adaption updating algorithm as depicted in Fig. 1. The whole adaptation framework includes two main parts: user feedbacks and adapting user preference profile by analyzing the feedbacks. This chapter discusses this dynamic adaption model.

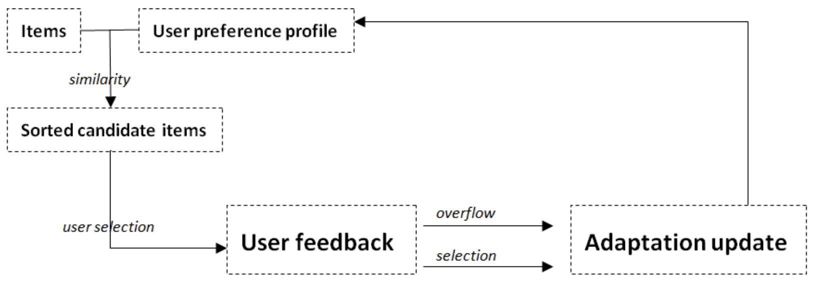


Fig. 1 Dynamic Adaptation Updating System architecture

3.1. User feedback

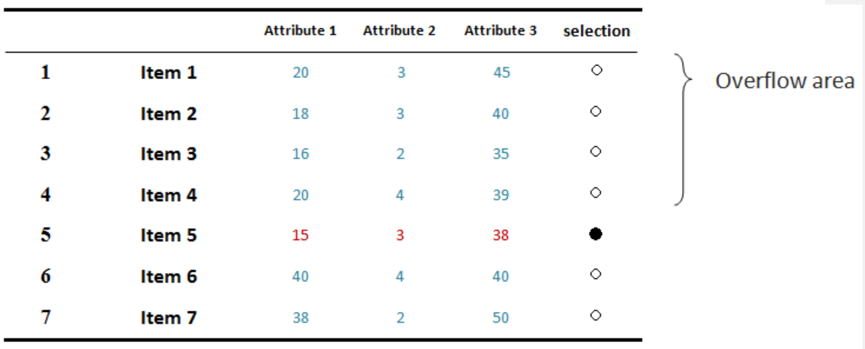
When the sorted candidate items are given to the target user, the next process is user selection. The favorite item selected by user is considered as user feedback.

User feedback consist of two pieces: the selected item and ranked-above 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 indirectly reflects user's real preference. Therefore, it can be used to update user preference profile. Moreover, items ranked above the selected ones are also significant, which are important to update user preference profile. These items are not selected by user, however, they are ranked in the front of the list. Therefore, the influence of these items should not be ignored.

Commented [GX2]: Overflow area doesn't make a clear sense

The example shown in Table 1 represents how a user selects an item from the sorted alternatives.

Table 1. Recommended items and User Selection



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

Overflow area

3.2. Model of dynamic adaptation

Commented [GX3]: ?

4.2.1 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". We assume that there is an attraction force between user preference profile and selected item, because the selected item reflects user's preference to a certain extent. Those items that are not selected but ranked among the top few will repulse user preference profile. Moreover, when selecting an item with a few attributes, we pay more attention to only one or two attributes. If this situation is considered, the values of user preference profile can be attracted and pushed away to the better values, which are closer to the ideal preference of the target user.

Commented [GX4]: Not sure what do you mean, tuned?

4.2.2 Traditional adaptation process

As stated above, the adaptation was inspired by Coulomb's Law [2, 32]. The values of user preference profile are attracted by attraction force between 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 the definition, the attraction force of attribute *i* will be defined as:

Commented [GX5]: ???

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

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 with attraction force, that is:

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

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

$$F^i = \theta \times F_a^i + F_r^i.$$

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:

- Items in the overflow area make different contributions to repulsion force. (in order to improve the algorithm performance)
- When users select an item with a few attributes, they only pay attention to one or two attributes. (in order to agree with the actual situation)

4.2.3 Influence of the overflow area

Commented [GX6]: Refer to above

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 are analyzed in detail.

Take Table 1 for example, we have four items in the overflow area. Item 3 is very similar with the selected item (item 5). It reminds us that the item 3 makes the least contribution in computing repulsion force. Then, a mathematical hypothesis is proposed that different items will contribute differently in computing the repulsion force. That means different over-ranked items should be assigned different proportions. 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 of 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 repulsion force caused by over-ranked item. It is noticed that 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 the whole repulsion force is the least. Based on the above discussions, a logit function [7] is proposed to compute the probabilities.

Commented [GX7]: ???

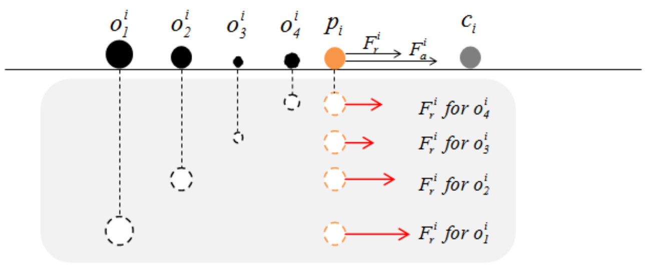


Fig. 2 repulsion forces of different over-ranked items

Firstly, every attribute *i* of each over-ranked item is compared with attribute *i* of selected item, then we achieve a new ranking vector of every attribute i of each over-ranked item. It is defined as $R^i = \{r_1^i, r_2^i, \ldots, r_m^i\}$. 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 $[-\infty, +\infty]$:

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

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

$$P_j^i = \frac{\exp(\hat{r}_j^i . r_j^i)}{\sum_{j=1}^{m}\exp(\hat{r}_j^i . r_j^i)} \in [0,1] \quad , \tag{3}$$

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

$$P_j = \frac{\sum_{i=1}^{n} P_j^i}{n} \quad , \tag{4}$$

where n stands for the total number of attributes. Then these probabilities are used to control the process of computing repulsion force. Pseudo-code in Fig. 3 has shown the proposed algorithm.

```
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

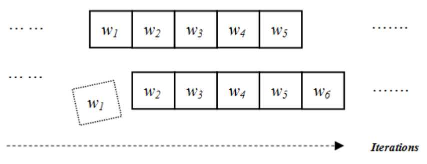
4.2.4 Weights of different attributes

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 weights of different attributes to provide 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 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 more and more accurate, the distance of every attribute between user preference profile and user selection will be increasingly closer. The weight of attribute *i* is learned by:

$$w_i = (1 - \frac{|c_i - p_i| / \Delta_i}{\sum_{i=1}^{n} |c_i - p_i| / \Delta_i}) / n - 1 \quad , \tag{5}$$

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].



w_1 w_2 w_3 w_4 w_5
w_1 w_2 w_3 w_4 w_5 w_6
Iterations

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

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:

$$W_i = \frac{1}{s} * \sum_{t=1}^{s} w_i^t \quad , \tag{6}$$

Pseudo-code in Fig. 5 has shown the proposed algorithm of computing weight of each attribute.

```
INITIALIZATION
    C(c^1,..., c^n),    // the selected item
    P(p^1,..., p^n),    // user preference profile
    k,    // size of moving window
    △,    // 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

4.2.5 The whole process of dynamic update of user preference profile

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

$$F_a^i(p_i, c_i, w_i, \omega) = \begin{cases} \dfrac{\Delta r_i.(c_i - p_i)}{|c_i - p_i|^{\omega}}.W_i & \text{if } c_i \neq p_i \\ 0 & \text{if } c_i = p_i \end{cases}, \qquad (7)$$

$$F_r^i(p_i, P_j, \{c_i^j\}, k, \omega) = \begin{cases} \displaystyle\sum_{j=1}^{k} \dfrac{(p_i - c_i^j).P_j.\Delta r_i}{|p_i - c_i^j|^{\omega}}.W_i & \text{if } c_i \neq p_i \\ 0 & \text{if } c_i = p_i \end{cases}, \qquad (8)$$

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*, it can be achieved by E.q. (6). Then , the final force of attribute *i* will be calculated by:

$$F^i = \theta \times F_a^i + F_r^i \qquad (9)$$

The whole process of dynamic update of user preference profile is depicted in Fig.

6.

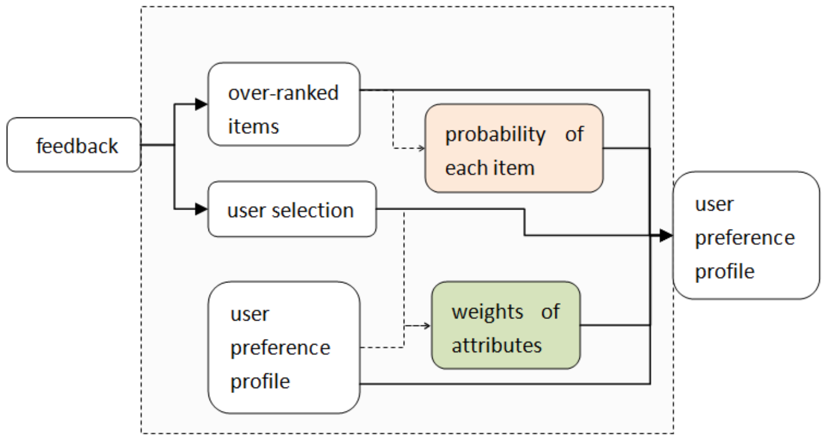


Fig.6 Dynamic update of user preference profile

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. (1) 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. Eq. (1) will be used in this step.
- Collecting user feedbacks (over-ranked items and the selected item).
- Updating user preference profile using Eqs. (7)-(9).

5.1 Data preparation of simulation

The data set of this problem was generated according to the "OpenAPI" of Ctrip (www.crip.com, booking restaurants) and explanations in [8]. Based on the description of restaurants, restaurants datasets are evaluated by 5 numerical attributes: "price", "distance to city center", "distance to airport", "room size and star". Table 2 shows the range of five attributes. A group of 1500 restaurants with five numerical attributes are generated randomly with a uniform distribution based on the range.15 restaurants are ranked independently, providing a total of 100 different recommendations.

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 2 Range of five attributes

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

	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

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

Take the student group as an example. A student prefers attribute v_1 (price) when selecting the item through the RS, we therefore define the weight of attribute one as 0.8 and the others defined as 0.05 (total is 1) in order to simulate different weights of the mind of a real user. As for the businessman, comprehensive experience is his/her concern major concern. For traveler, he/she prefers traveling convenience. Different types of users have their unique preferences. The initial weight of each attribute is set to 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 dynamic adaptation are: (1) moving the values of user preference profile closer to user's ideal preference. (2) finding 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 overflow area. 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

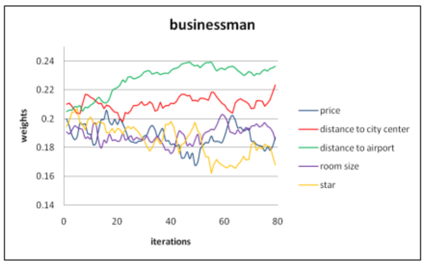
Previous sections have presented a useful model aiming at updating user preference profile by combining the significance of different over-ranked items and weights of different attributes. This section describes the results obtained from experiments. All results shown in this section are the results averaged 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

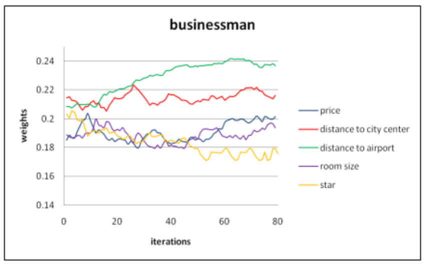
Figures 11 shows the changes of weights of different attributes with various window sizes. From the evaluations results, we can get two important conclusions:

- The most important attributes can be found immediately after several interactions. Take student for example (Fig. 12(b)), his/her most interested attribute is price. It is seen that the weight value of attribute price rises from 0.2 to 0.25, while the other attributes decline. Meanwhile, the most concerned attributes of businessman (distance to airport) and traveler (distance to city center) are discovered similarly.
- 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.

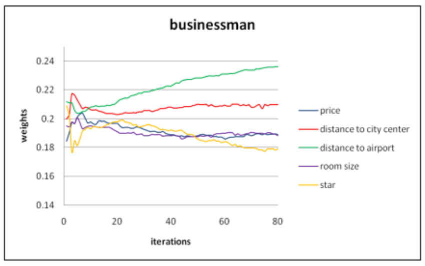
Commented [GX9]: The weight value varies continuously, does it converge to a stable state? Can't see from the curve



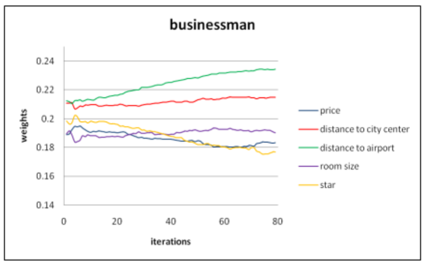
window size=0



window size=3

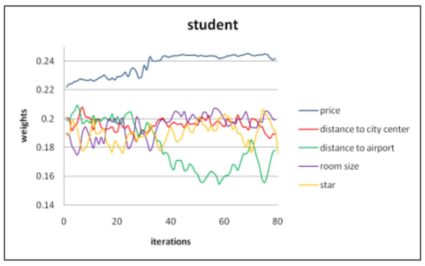


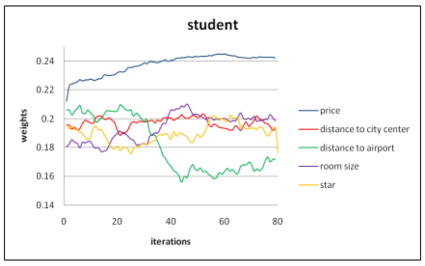
window size=5



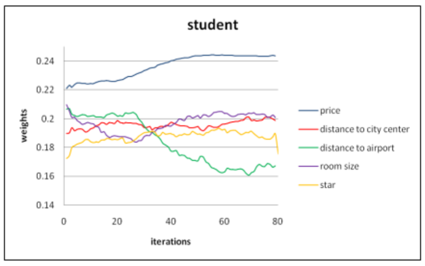
window size=8

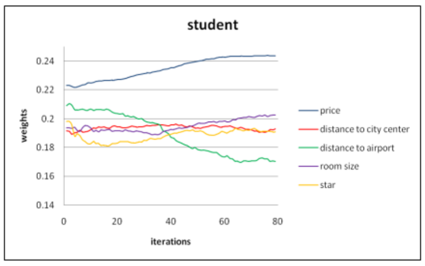
(a) - businessman





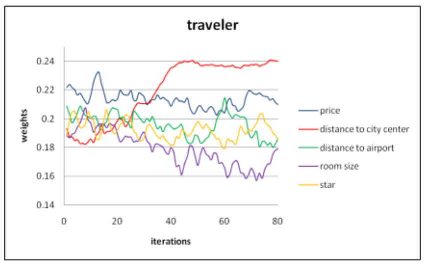
window size=0 window size=3

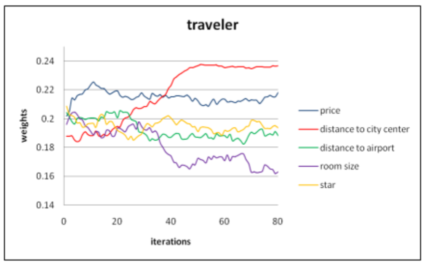




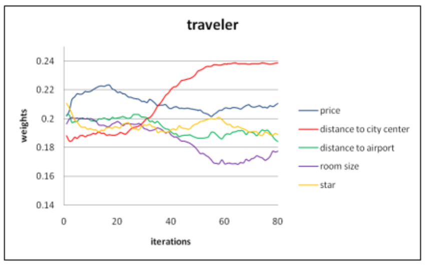
window size=5 window size=8

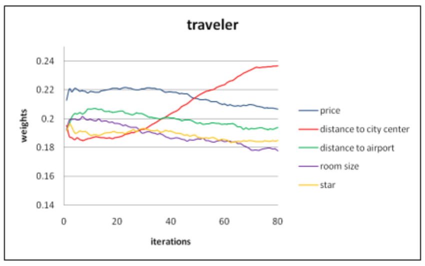
(b) - student





window size=0 window size=3





window size=5 window size=8

(c) - traveler

Fig. 11 Evaluation of the weights of different attributes with different window sizes

Commented [GX10]: ??? the figure numbers should be ordered

5.3.2 Evaluation the result of updating user preference profile

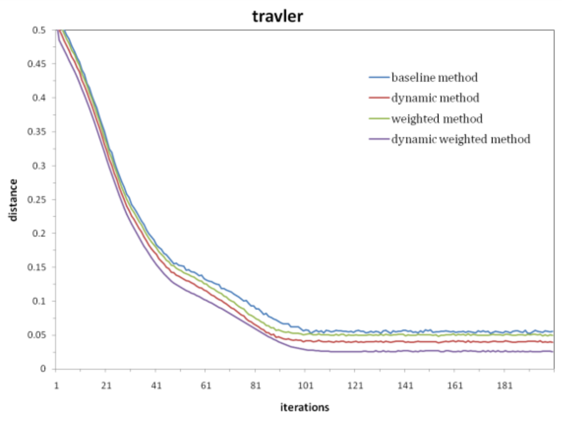
In order to evaluate the results of the new learning model, a distance function is used to calculate how different the user preference profile to be updated is for an ideal profile, which represents the exact preferences of the target user. The distance between two profiles is computed by:

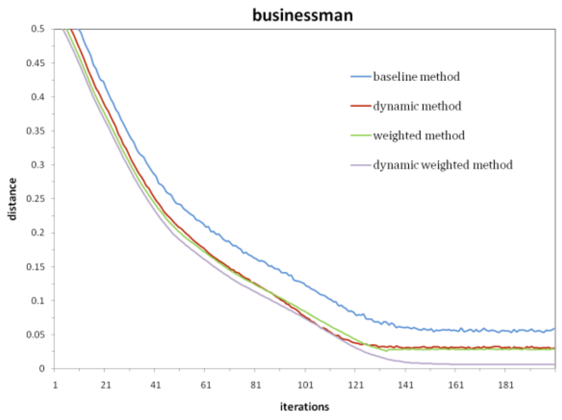
$$d(p,c,\Delta)=\frac{1}{n}\sum_{i=1}^{n}|\,p_i-c_i\,|/\Delta_i\ ,$$

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

Fig 7 compares our method to the other three methods for three types of users, the results show that our method has improved the result of previous ones. It comes the conclusion from the experiment that there is no obvious difference between the results of weighted method and dynamic method to the businessman. The reason of it is that when a businessman is selecting hotel, he prefers considering every weight of attributes simultaneously —different weights of attributes don't make influence on his choice. On the other side, to the students and travelers, the result of weighted method is better than the result of dynamic method. They prefer considering some attribute (student---price, traveler---distance to city center) when choosing hotel, and that is what makes a bigger influence on the choice. For example, the preference of student choices is prone to lower and economical prices. The result and analysis above are consistent to the characters of the ideal profile of such three types of people in real circumstance. What is more, the result is 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).

Commented [GX11]: ???





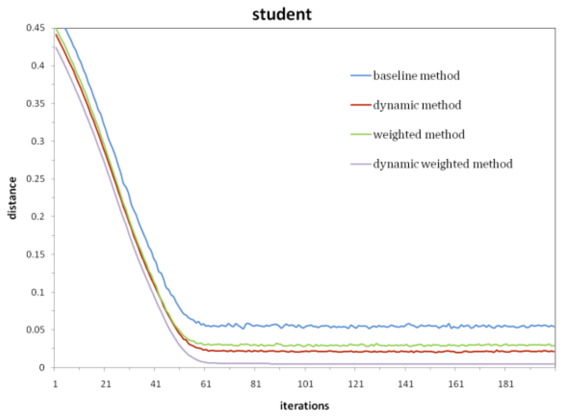


Fig 7. Average distance between the current and the ideal profile

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. 8 shows the position of the ranking performed by the RS where the user selection is located. It can be seen 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 is ranked in the first position.

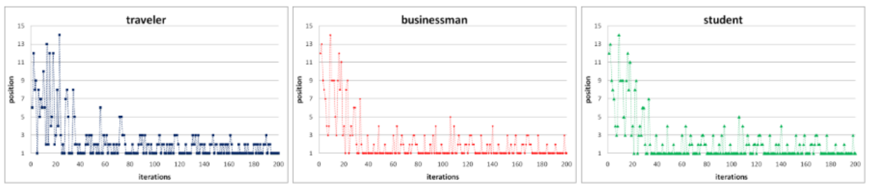
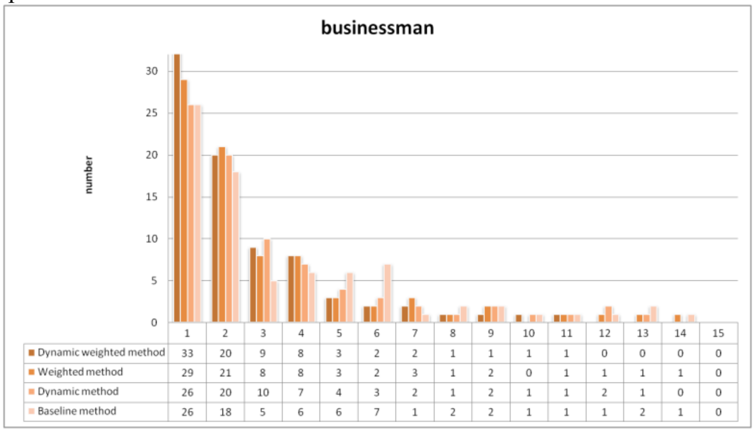
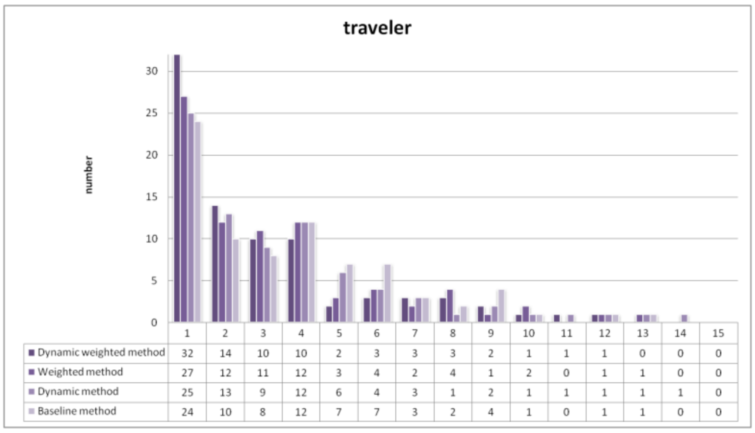


Fig 8. Ranking position of user's favorite item of recommendations.

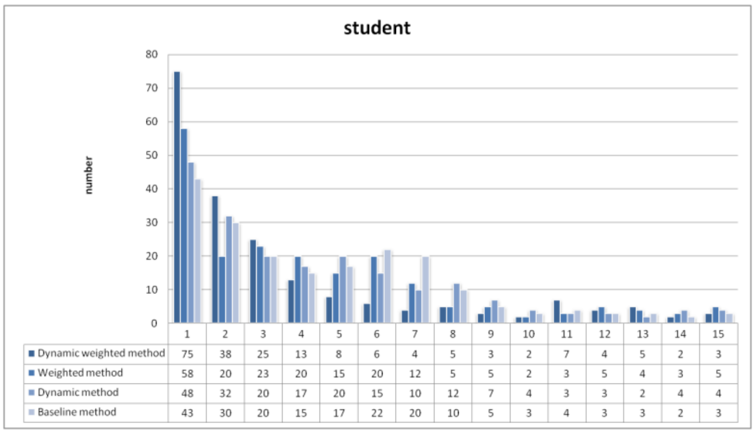
Fig. 9 shows the results of comparison between the three baseline methods and proposed method. Y-axis stands for the user selection's position (the item selected by user and its recommended position) among the 15 recommendations. The table behind it selects the statistic number of each user selection's position during 200 iterations. For the front recommended items (position 1,2,3) , our method collected much more numbers than the other methods, which means the performance of updating the profile has improved.



	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 9. 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.

Additionally, as the profile is updated accurately, it does help to substantially improve the accuracy of the ranking of recommended list which is based on the profile for offering to the users. In other words, it is correct to memory user's preference profile as the recommended rank 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 metric of Normalized Discounted Cumulative Gain (NDCG), 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

Commented [GX12]: ???

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

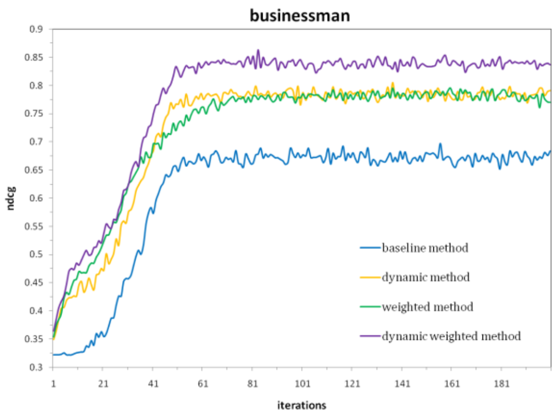
$DCG_{k\text{-}ideal}$ (ideal DCG) is the best DCG that we could obtain. $DCG_{k\text{-}ideal}$ is obtained by analyzing the ideal ranking list, which will match the ideal user preference. DCG is obtained by analyzing the recommended ranking list. DCG is defined as:

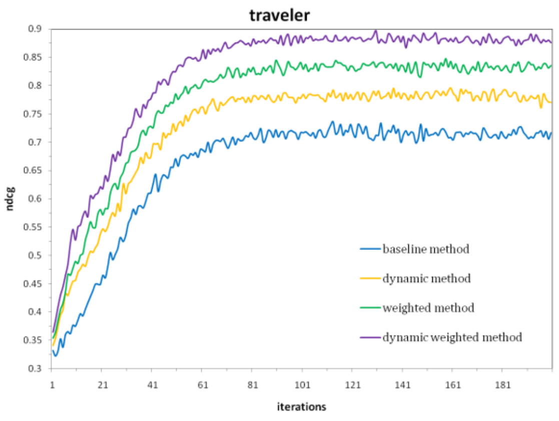
$$DCG_k = \sum_{i=1}^{k} \frac{(2^{s(i)} - 1)}{\log_2(1 + \mathrm{i})}$$

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, candidate at the first position of ideal ranking list gets 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 one.

Figure 10 shows the evolution of the $nCDG_{15.}$ As the updated user preference profile and the ideal profile get more close, nCDG$_{15}$ increases.

We can get a conclusion from the experiment that the nDCG result of dynamic method is almost the same as that of weighted method to the businessman. The results are both more superior to the baseline methods. To student and traveler , the nDCG result of weighted method is more worse than that of dynamic method. The reason of that is the same as the result came from the analysis of Figure 7. The result and analysis above are conistent to the characters of the ideal profile of such three types of people that the paper has defined. What is more, the nDCG result is better when it is not computed according to the other three methods after simultaneously considering the weighted factors and the over-ranked items (the model the paper has proposed). To the businessman, according to the model in paper ,the ndcg is approximately 0.85 and the baseline method is only about 0.68.To the travelers, the ndcg is approximately 0.87 and the baseline is only 0.7.To the students, the ndcg is approximately 0.78 and the baseline is only 0.59.





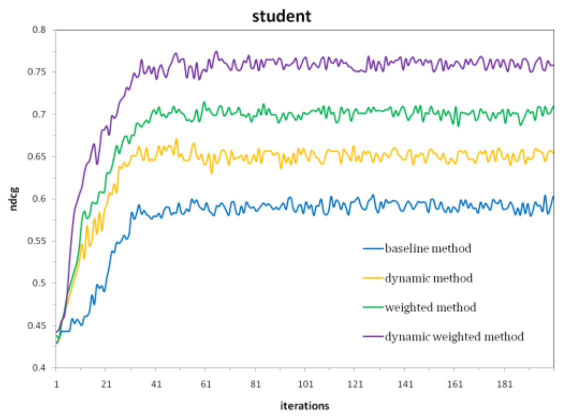


Fig 10. NDCG analysis performance

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. User preference profile stores user's numerical preference information, yet the initial user preference profile is not the most accurate profile and it is needed to be updated through interactions between user and 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 one, 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 propose a method to find the most concerned attribute(s). The results of simulation has proved that our method are able to enhance the performance of learning user profile, and the weighted algorithm helps us in finding the most concerned attributes.

The authors hold that the following four aspects are suitable targets for further study:

- The semantic attributes (comments) and categorical attributes (very big, big, small, very small) could be added into the proposed model. That means a more appropriate method should be created to represent the 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 could be achieved for future study.
- Combining the proposed model with a more efficient recommendation algorithm. The excellent recommendation algorithm is the basis of 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 the user numerical preferences.

Acknowledgement

This work has been supported by the Fundamental Research Funds for the Central Universities 2016YJS028 and the National Natural Science Foundation of China under Grant 61172072, 61401015, 61271308.

References

1. Lika, B., K. Kolomvatsos, and S. Hadjiefthymiades, *Facing the cold start problem in recommender systems.* Expert Systems with Applications, 2014. **41**(4): p. 2065-2073.
2. Marin, L., A. Moreno, and D. Isern, *Automatic preference learning on numeric and multi-valued categorical attributes.* Knowledge-Based Systems, 2014. **56**: p. 201-215.
3. Briguez, C.E., et al., *Argument-based mixed recommenders and their application to movie suggestion.* Expert Systems with Applications, 2014. **41**(14): p. 6467-6482.
4. Abbas, A., L.M. Zhang, and S.U. Khan, *A survey on context-aware recommender systems based on computational intelligence techniques.* Computing, 2015. **97**(7): p. 667-690.
5. Adomavicius, G. and D. Jannach, *Preface to the special issue on context-aware recommender systems.* User Modeling and User-Adapted Interaction, 2014. **24**(1-2): p. 1-5.
6. Adomavicius, G., et al., *Context-Aware Recommender Systems.* Ai Magazine, 2011. **32**(3): p. 67-80.
7. Walter, F.E., S. Battiston, and F. Schweitzer, *A model of a trust-based recommendation system on a social network.* Autonomous Agents and Multi-Agent Systems, 2008. **16**(1): p. 57-74.
8. Marin, L., A. Moreno, and D. Isern, *Automatic learning of preferences in numeric criteria.* Artificial Intelligence Research and Development, 2011. **232**: p. 120-129.
9. Zhang, Z., Y., Liu Y., Qing A Z., *A Weighted Method to Update Network User Preference Profile Dynamically.* International Journal of Interdisciplinary Telecommunications & Networking, 2014. **2**(6): p. 52-67.
10. Adoinavicius, G. and Y.O. Kwon, *New recommendation techniques for multicriteria rating systems.* Ieee Intelligent Systems, 2007. **22**(3): p. 48-55.
11. P. Li, C.J.C.B., 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,*2007: British Columbia, Canada.
12. Ifada, N. and R. Nayak, *Do-Rank: DCG Optimization for Learning-to-Rank in Tag-Based Item Recommendation Systems.* Advances in Knowledge Discovery and Data Mining, Part Ii, 2015. **9078**: p. 510-521.
13. Jin, J., et al., *Learn to Rank: Automatic Helpfulness Analysis of Online Product Reviews.* Proceedings of the Asme International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Detc 2010, Vol 3, a and B, 2010: p. 467-476.
14. Banati, H. and M. Bajaj, *Dynamic User Profile Generation by Mining User Opinion.* Third International Conference on Computer Engineering and Technology (Iccet 2011), 2011:

p. 557-562.

15. Chen, L., G.L. Chen, and F. Wang, *Recommender systems based on user reviews: the state of the art.* User Modeling and User-Adapted Interaction, 2015. **25**(2): p. 99-154.

16. Rokach, L. and S. Kisilevich, *Initial Profile Generation in Recommender Systems Using Pairwise Comparison.* Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews, 2012. **42**(6): p. 1854-1859.

17. Marin, L., et al., *On-line dynamic adaptation of fuzzy preferences.* Information Sciences, 2013. **220**: p. 5-21.

18. M. Pazzani, D.B., *Learning and Revising User Profiles: The Identification of Interesting Web Sites.* Machine Learning, 1997. **27**(3): p. 313-331.

19. K.-Y. Jung, K.-W.R., 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*2004: Lisbon, Portugal. p. 815-823.

20. Morales-del-Castillo, J.M., et al., *Recommending Biomedical Resources A Fuzzy Linguistic Approach Based on Semantic Web.* International Journal of Intelligent Systems, 2010. **25**(12): p. 1143-1157.

21. M. Montaner, B.L., J. L. de La Rosa, *A taxonomy of recommender agents on the internet.* Artificial Intelligence Review, 2003. **19**(4): p. 285-330.

22. L. Marin, D.I., A. Moreno, *A Generic User Profile Adaptation Framework*, in *13th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2010*2010: Tarragona, Spain. p. 143-152.

23. Chen, T., et al., *Content recommendation system based on private dynamic user profile.* Proceedings of 2007 International Conference on Machine Learning and Cybernetics, Vols 1-7, 2007: p. 2112-2118.

24. Haruechaiyasak, C., et al., *A dynamic framework for maintaining customer profiles in e-commerce recommender systems.* 2005 Ieee International Conference on E-Technology, E-Commerce and E-Service, Proceedings, 2005: p. 768-771.

25. Schwab, I.P., W. Koychev, I, *Learning to Recommend from Positive Evidence*, in *Proceedings of Intelligent User Interfaces 2000, ACM Press*2000. p. 241-247.

26. Peis, E., E. Herrera-Viedma, and J.M. Morales-del-Castillo, *A semantic service model of selective dissemination of information (SDI) for digital libraries.* Profesional De La Informacion, 2008. **17**(5): p. 519-525.

27. Morales-Del-Castillo, J.M., et al., *A Semantic Model of Selective Dissemination of Information for Digital Libraries.* Information Technology and Libraries, 2009. **28**(1): p. 21-30.

28. Sebastia, L., et al., *e-Tourism: a tourist recommendation and planning application.* 20th Ieee International Conference on Tools with Artificial Intelligence, Vol 2, Proceedings, 2008: p. 89-96.

29. L. Baltrunas, X.A., *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*2009: New York, USA. p. 25-30.

30. E. Cheng, F.J., 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*2006: Toronto, ON, Canada. p. 1773-1776.

31. Veilumuthu, A. and P. Ramachandran, *Discovering implicit feedbacks from search engine log files.* Discovery Science, Proceedings, 2007. **4755**: p. 231-242.

32. Marin, L., D. Isern, and A. Moreno, *Dynamic adaptation of numerical attributes in a user profile*. Applied Intelligence, 2013. **39**(2): p. 421-437.