Chapter 5
A Multi-Criteria Recommender System Based on Users' Profile Management

Arnaud Martin, Pascale Zarate, and Guy Camillieri

Abstract The work consists in developing a recommender system in order to support decision makers in their activities. This support is possible through the management of users' profiles that will evolve following their answers or actions. This evolution is possible using automated techniques, especially reinforcement learning. The developed recommender system is based on a Multi-Criteria approach.

5.1 Introduction

Considering user profiles and their evolutions, is considered in the Decision Support Systems (DSS) community as an important issue [8]. Indeed, the inclusion of context in the decision is currently emerging for DSS. The purpose of a DSS is to support a decision maker giving him solutions or parts of solutions to his problem. A natural evolution of DSS was to offer advices to users based on their profile. These profiles generally represent decision makers' preferences. Several approaches are possible to define these profiles. One of them consists to determine a list of valued criteria. The criteria are defined according the decision making problem to solve. It is called Multi-criteria Recommender System. The main challenge for Recommender Systems comes from the fact that the system needs to continuously bring relevant information. It therefore requires changing user profiles thanks to their actions. So, the system must not only "understand" what the user likes, but also why. The users' assistance will evolve over the time and therefore with the user. Thus the user has at his disposal a kind of personal assistant.

A. Martin
LIGANZ, Nice, France

P. Zarate (✉)
IRIT (Institut de Recherche en Informatique), Toulouse University, Toulouse, France
e-mail: Pascale.Zarate@ut-capitole.fr

G. Camillieri
IRIT (Institut de Recherche en Informatique), Toulouse University, Toulouse, France

C. Zopounidis, M. Doumpos (eds.), *Multiple Criteria Decision Making*, Multiple Criteria Decision Making, DOI 10.1007/978-3-319-39292-9_5

The objective of this work is to provide assistance to decision makers' activities according to their profiles. The objective is to develop an algorithm based on automatic learning techniques, in order to allow the profile evolution. Then, the provided support will dynamically change with the user's profile changes. In order to achieve this objective, a refining is performed through scalable scheduling solutions presented to the user depending on his/her profile. Nevertheless, the users' inputs must be considered at two levels: in one hand items chosen by the users and in another hand, actions done by users. As an output, the system provides a ranked list of items.

In the Sect. 5.2, we define the recommender systems. Then in the following section, Sect. 5.3 Multi-Criteria DSS are described. The next section, Sect. 5.4, is devoted to machine learning description. The Sect. 5.5 gives a general view of the developed system. The next section, Sect. 5.6, proposes a detailed description of the system operating. The Sect. 5.7 allows us to show how the implemented system is correctly working through several tests and results. In this section frameworks have been developed to evaluate the system and compare it to other systems. Finally, several conclusions are given and some perspectives are drawn.

5.2 Recommenders Systems

Users have no longer time to look at all available information, that's why the recommender systems can be extremely useful to direct and filter information that users have to access. [28]

These systems rely on user profiles representing their preferences, and to filter, order information and select the best information fitting to users' preferences. The general process of recommender systems is decomposed in three steps:

1. The user provides a list of examples of his tastes, which can be explicit, such as notes on specific elements, or implicit, as visited URLs, or selecting an object from a list.
2. These data are used by the system to create a representation of the user's profile, what he likes or does not likes, and how much.
3. The system computes recommendations according to the user's profile.

The proposed solutions are actually the following:

- Synchronous solution: adaptive factors (criteria's score) are derived from the filtered items accumulated over-time [15].
- Asynchronous or deferred solution: factors are derived from existing collections of objects. These collections also provide examples of relevant elements for each profile.

The synchronous solution offers the possibility of having a system that evolves in real time. Thus the user has a direct feedback of his actions. However, this solution requires the use of appropriate computations in their complexity, implying their cost/time computation, compared to the expected reactivity of the desired system. This is major constraint for DSS as it has been proved that in order to facilitate the cognitive following of the problem solving the answering time must be as little as possible (no more than 3–4 s).

The asynchronous solution eliminates the problems of cost computation, but the user's experience is reduced in quality. Indeed, these systems update user's profile and perform various other computations in a deferred way.

Several methods, taking into account the user's past, have emerged to recommend items to users:

- Social recommendation (collaborative filtering): make recommendations based on past behavior of similar users [3, 7]. Users are here categorized in classes and recommendations are made following the categories.
- Recommendation Object (content based): recommend things based on the intrinsic qualities of the object itself [18, 30]
- Hybrid recommendation: a combination of the two approaches described above [1, 26]

Many systems use collaborative filtering; these systems can satisfy a lot of people who do not have very different tastes. But they cannot satisfy users with precise tastes which are not very popular, and they do not work properly with few users [9]. Moreover with these systems, new items or not yet rated items may be discarded in the process of recommendation and the new objects are not recommended.

In another hand, content based systems can rank all items without taking into account other users. But the main problem remains to have a sufficiently precise description of these items, and to be able to handle all these data.

A multi-criteria approach including a database describing precisely items allows us to define an efficient recommender system.

5.3 Multi-Criteria Decision Support

The practice of using multiple criteria decision often leads to build recommendations more varied than just choosing one and only one action [23]. That's why, in order to better take into account user preferences, we propose to use multicriteria/multiattribute systems. This allows us to better manage and use of user preferences.

To properly do the difference between an attribute and a criterion, we need to introduce the following definitions:

- Attribute: Characteristic describing each objects (age, qualifications, aptitude test results, claims).
- Criterion: Expresses preferences of the decision maker with respect to a point of view (eg powerful car). This concept incorporates the preferences of the decision

maker on this criterion (maximizing power, maximum speed). A criterion may refer to one or more attributes. For example, the criterion of "skill" refers to attributes such as qualifications, past experiences, etc.

When some subjective characters related to the decision maker preferences are introduced into the description of an object, the word "criterion" is gladly more used. However, we often can define a criterion by a single attribute and generally the following approximation is done: ***attribute*** $\leftrightarrows$ ***criterion***. To resume the criterion of "skill", it can be defined, for example, as a single attribute corresponding to a skill level, computed from other attributes.

That's why, generally, and for the rest of this work, we use the following approximation ***attribute*** $\leftrightarrows$ ***criterion***. Two "schools" exist in Multi Criteria Decision Aiding (MCDA) and follow quite different basic principles [16]. The first approach is the "American School", which most often uses an additive utility function that combines the utility values in an overall score for the action. The simplest aggregation method of this category is the weighted sum, where the overall rating is the weighted sum of scores for each selected criterion multiplied by its weight. Other methods of this type are MAUT, Multi Attribute Utility Theory MHM, multicriteria ranking method or AHP Analytical Hierarchy Process [25].

The "European school" promotes methods based on comparisons between potential actions. The methods set which are the best known, are the ELECTRE (Elimination Et Choix Traduisant la Ralit) methods [21, 22, 24] and the PROMETHEE method, Preference Ranking Organization Method for Enrichment Evaluations [6]. [5]

There are still many other methods that do not belong to one or other of the two "schools" as Qualiflex [17] and methods using the principles of cost and benefit [19]. All these methods require obtaining an a priori weight and other parameters.

It is generally accepted that "a priori" preference of the decision maker is very difficult to obtain, which explains the development of the interactive methods aiming to progressively obtain preferences [10]. An interactive method consists in alternating steps of computations and steps of dialogue with the decision maker. The first stage of computations provides a first solution. It is presented to the decision maker, and he (or she) reacts by providing information about its preferences (dialog step). This information is injected into the model and allows building a new solution [29].

Interactivity also allows the decision maker to better understand their preferences regarding the addressed problem, and change them based on a gradually improved understanding. In the same time this approach facilitates accepting the recommendations of the system, since the decision maker can continue the dialogue until he (or she) is satisfied [10]. This kind of methods can be easily coupled with machine learning technics in order to satisfy the users as much as possible.

5.4 Machine Learning

The machine learning refers to the development, analysis and implementation of methods that allow a machine, with a large meaning, to evolve through a learning process. These technics allow fulfilling tasks which are difficult or impossible to perform in ways of more conventional algorithms.

Several methods exist but the most interesting method for our work is the so-called reinforcement, because it is particularly suitable for profiles updating. The method of reinforcement learning applied to user profiles has been presented and most experienced in [28] and [4].

This method was used in a system that was performing information retrieval as well as information filtering. It is described as follows: "Learning profiles that we use is based on the principle of reinforcement [27]. To this end, we consider $wo_{ik}^{(t)}$ corresponding to the weight of attribute k of the selected object $o_i^{(t)}$ and considered relevant. We must find a representation of the profile $p_x^{(t)}$ at the time t which retrieves this object with a "strong" score λ. Then use these data in order to update the overall profile of the user. We must find $wc_j^{(t)}$ as weight of the attribute a_j in the profile $p_x^{(t)}$, which satisfies the following equation":

$$\sum_{k \in o_i^{(t)}, aj \in p_x^{(t)}} wo_{ik}^{(t)}.wc_j^{(t)} = \lambda$$

For integration into the user profile, the formula for distribution of the gradient is used as follows:

$$w_i^{t-1} = w_i^t + 0.1 * \log(1 + wc_i^{(t)})$$

- w_i^{t-1} is the weight of criterion i at time $t - 1$
- w_i^t is the weight of criterion i at time $t - 1$
- $wc_i^{(t)}$ is the weight of criterion i at the current object

This method is very suitable for our work thanks to the introduction of several criteria in the profile definition. However, there are many limitations for this method, in particular the way how the intermediate weights are computed. Indeed, the system proposed by Boughanem et al. [4] aims to provide an information filtering system, and use it only for textual data. Thus the way how the $wc_j^{(t)}$ are obtained cannot be used because it refers only for text types data. In our case, the types of used data are diverse, and vary depending on objects and their description. We provide an adjustment in order to adapt these formulas to our goal and our data types.

5.5 System Description

Our aim is to develop a system able to recommend on several kinds of items or objects. Our system focus on a content based approach.

We firstly define, build and allow the evolution of a user profile (evolutionary profiling) based on explicit and implicit user's actions. This evolutionary profiling is implemented within a recommender system usable without learning base, i.e. initialization, synchronously and completely incremental. The system is open for inconsistencies because this work is based on Bounded Rationality that is one hypothesis of the decision making domain. This system, which complements an Information System Research, aims to establish a total order on a list of items proposed to the user in accordance with his preferences. The development of such a system implies the following challenges:

- The disaggregation of criteria
- The inclusion of a variable number of criteria.

We firstly define a coherent family of criteria on which the decision is based. This family of criteria is the total set of criteria necessary to evaluation an item. All the criteria are not necessarily valued at each step of the solving process.

The built system complements retrieval system information. Indeed, as described in [2], information filtering (and recommender systems) is a dual process to the information retrieval. So the goal of our system is to order the available solutions for the user by taking into account the user's profile.

The user's profile is composed by a vector of weight criteria representing the user's preferences. We want to learn the user's profile which implies to obtain these weights. In addition, we want those weights without asking directly them.

Several constraints for designing the system have been defined. Thus, the system will be used by various users, for who we don't know anything. Indeed, for the initialization phase, we do not have basic information about the user and therefore we cannot use algorithms using a learning base.

In addition, users have the ability to change preferences even if they are not rational (bounded rationality).

Another constraint is to retrieved information about selected items or rated items by the users in a sequential manner. So we need to update the user profile, each time data are collected. We have to learn continuously (there is no learning delay), in synchronous way and with an acceptable response time.

The system must have the ability to be used for any type of items (items such as books, telephones, houses, etc …). We cannot use the approaches already developed, for example related to the recurrence of a word in a text [20].

We propose a purely incremental method of profile learning, which requires no knowledge to start the recommendation process.

This method begins first to build a temporary vector of valued criteria based on a set of criteria describing a rated or selected element by the user. This temporary profile is then integrated into the overall profile of the user.

We provide to the user recommendation based on a ranking of proposed items, using a scoring method to obtain an overall score for an item. This score is calculated for the evaluated item from its characteristics and the user's profile.

5.6 System Operating

5.6.1 Introduction

The first step is an initialization step for new user, i.e. for who we don't have any information. In this case, we provide a random list of item to him, and all criteria weight in his profile are set to 0. Otherwise, the system begins by present a list of item to the users, this list is ranked by his profile and his preferences, better items first, if we already have information about the user. In a second step, these are two possible actions for the use: select an item or rate an item.

The system uses is described in the following Fig. 5.1.

1. User begins by doing an action on a presented item. Indeed, if he already knows an item in the list, he rates this item (explicit data). Otherwise, if he does not know any item in the list, he looks the characteristics of items and selects the one he prefers (implicit information).
2. Depending of his choice, a Rate computation, i.e. explicit data, or a Pairwise computation, i.e. implicit data, is launched in order to disaggregate the item score.

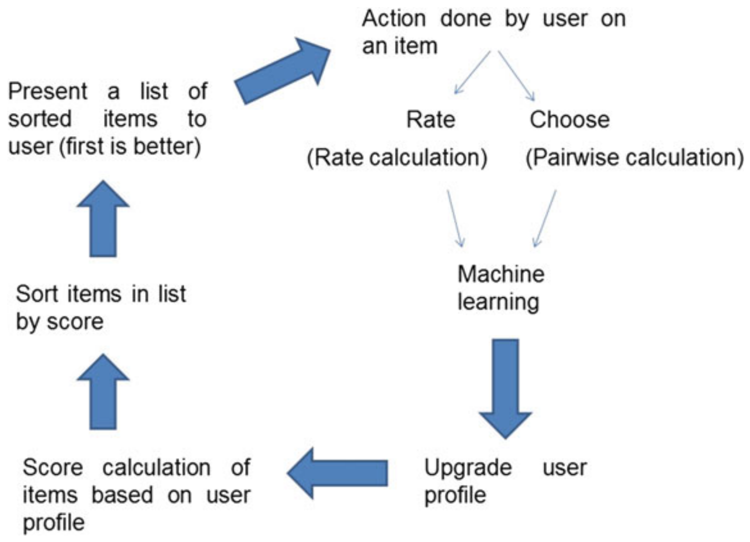
Action done by user on an item
Rate
(Rate calculation)
Choose
(Pairwise calculation)
Machine learning
Upgrade user profile
Score calculation of items based on user profile
Sort items in list by score
Present a list of sorted items to user (first is better)

Fig. 5.1 System operation schema

3. With these data the system computes a machine learning algorithm and updates the user profile.
4. The system makes a score computation of available items, for the global database or a selection obtained through a request, based on user profile.
5. The system sorts items with these new scores.
6. The new list is presented to the user (this list is reduced to the top ten items).

5.6.2 Scoring

For each element we compute a score (Item Score) which represents the satisfying value of an item for the user. This score is used to rank the list of presented items to the user (descending order). As used in the reinforcement learning, we use a sum-type function which is based on the utility principle.

We calculate the item score as follows:

$$\text{Item Score} = \frac{\sum_{i=1}^{N} s(t)_i}{N}$$

- N corresponds to the number of criteria in item
- $s(t)_i$ corresponds to the score of criterion i at time t in user profile

This sum corresponds to the scores contained in the user profile corresponding to the criteria contained in the element i at time t. We than are faced to two cases: if the criteria is not present in the user profile then add 0 to the score; if the criterion is present then add his score to the overall score. Example: An object defined by the following criteria: Brand, Age, Speed, Color A corresponding item can be: Ford, Modern, Fast, Red A user profile can be: (Ford, 9), (Fast, 3), (Red, 6)

The item score is then computes as follows: Item Score= $\frac{9+3+6}{3} = 6$

We then defined two methods to compute the score sc_i for the criterion i for the current item. This sc_i represents the user interest for this criterion i. These two methods are defined depending on the user action. The first method is based on the explicit rate and the second on the implicit information.

5.6.3 Rate Computation

In this case for which the user provides a rate on an item, scaled from 0 to 5, we got explicit data.

We calculate the distribution of scores for each criterion, and we integrate the resulting criteria vector in the user profile.

This is the way how we disaggregate the global value:

$$sc_i = \frac{\text{rate of item}}{N}$$

- sc_i corresponds to the score of criterion i at time t in user profile

One hypothesis of our work is that we consider that no criterion is more important than another. All the criteria of one item get an equivalent score.

Then, with these scored criteria, we are able to update the user profile.

In order to do this, we use the following formula:

$$s(t+1)_i = s(t)_i + 0.1 * \log(1 + sc_i)$$

- $s(t+1)_i$ is the score of criterion i at time $t+1$ in the user profile
- $s(t)_i$ is the score of criterion i at time t in the user profile. That is to say that $s(t+1)_i$ is the new score of criterion i and $s(t)_i$ is the former score criterion i in the user profile.
- sc_i is the computed score of the current element

We use a logarithm function in order to reduce the impact of high values. A coefficient of 0.1 is used to control the impact of new data on the system, and adding a parameter, i.e. 1, to the logarithm function in order to prevent the addition of negative values.

The second method to calculate the score sc_i for the criterion i for the current item based on implicit information is called pairwise computation and is described in the following section.

5.6.4 Pairwise Computation

We are here faced to the case where the user chooses an item. We launch a machine learning algorithm using a Pairwise computation. Several advantages to this type of methods are described in [12]. We use the following procedure: we construct a weight vector and we use it to update the user's profile.

We performed several intermediate computations, which are used to upgrade a temporary vector for each pair of items. These pairs of items consist in each case of a displayed item selected by the user and an item displayed above the selected one. Above corresponds, in this work, to the position of an element with a lower index in the list presented to the user than the selected one by the user. Indeed, more the item is located down in the list and greater the index is and more the Item Score is low.

We perform the computations by using the preferred object and alternately the above items, which make several pairs. This principle reflects the idea that for a sorted list, when the user selects an item that is not in first position, this corresponds to the fact that the user considers the selected item favorite to previous.

The temporary vector is used in order to storage the current calculation for the selected item.

The main process of the Pairwise Computation is described as follows:

1. Initialize temporary valued vector using a disaggregation rate for the selected item and an average rate, i.e. $Average(maxscale = 5) = 2.5$
2. Removing non-discriminating, i.e. criteria in common with the two items, criteria between the selected item and one above.
3. Do a sc_i computation, described here after, and upgrade the temporary vector with obtained data.
4. Repeat task 1 to 3, with another item above the selected one, until there is no other item above.
5. Upgrade user profile using the temporary vector.

In order to do the sc_i computation, we use the following method: The k item is considered as the one selected by the user, and the item j as an item above the selected one. We then have the following equation because item j is higher in the list presented to the user than item k:

$$\frac{\sum_{i=1}^{N} s(t)_{ij}}{N} > \frac{\sum_{i=1}^{N} s(t)_{ik}}{N}$$

- $s(t)_{ij}$ being the score of criterion i at time t for item j
- $s(t)_{ik}$ being the score of criterion i at time t for item k

But we want, because the user selected the item k that must be preferred to j:

$$\frac{\sum_{i=1}^{N} s(t)_{ij}}{N} < \frac{\sum_{i=1}^{N} s(t)_{ik}}{N}$$

This would imply that the built user's profile is correct. We then have to correct the user's profile. In order to correct it, we must determine which criteria and their related scores will update their profile. We then calculate the score difference between the two objects.

$$\Delta = \frac{\sum_{i=1}^{N} s(t)_{ij}}{N} - \frac{\sum_{i=1}^{N} s(t)_{ik}}{N}$$

Indeed, we have two items with their computed scores. We want that the selected item score higher than the score of the above item. The score difference (Δ) is then used for upgrading user's profile. We have used here an additive decomposition nontransitive by difference. We use Δ to get the score of each criterion included in our temporary vector.

$$SC_i = \frac{\Delta}{N}$$

Thus we can deduce that greater the score difference is, more the system is not properly adjusted. We also see that greater the difference is and more we need to correct the user's profile.

This correction is done by using Δ, because greater the difference is, greater Δ is, and more sc_i get an important value.

This allows us to adjust the system quickly. Indeed when Δ (and sc_i) have an important value, our updating have a large impact on the user's profile. With this computation we get a fast adaptive system which can handle users changing often their taste and therefore the weighting of criteria inside their profile.

Thanks to the sc_i we can update the user profile. We use almost the same formula that the one used in the rate computation for the upgrading. We add a variable who will allow us to have a fair impact on the system. Indeed, in order to maintain an effective system we want a fair impact on the system and that need to take into account the number of pair computation that we have done.

To do this, we use this formula:

$$s(t+1)_i = s(t)_i + 0.1 * \log(1 + \alpha * sc_i)$$

With: $\alpha = \frac{1}{\text{number of element above the one selected}}$

- $s(t+1)_i$ is the score of criterion i at time $t+1$ in the user profile
- $s(t)_i$ is the score of criterion i at time t in the user profile. That is to say that $s(t+1)_i$ is the new score of criterion i in the user profile and $s(t)_i$ is the former score criterion i in the user profile.
- sc_i is the computed score with the current element

Upgrading user's profile is then possible with explicit (rate) and implicit (selection by user) information. The implemented system must be tested and validated through several experiments described in the following section.

5.7 Experiments and Results

The developed system is implemented in order to validate the proposed calculation methods. For this purpose we need to compare it with existing systems. This comparison will be done through a free data set. We then searched a database with some constraints; indeed we wanted a multi-criteria database with rate from users and for each of them as much as possible marks.

We finally used the MovieLens dataset, which is a free service provided by GroupLens Research at the University of Minnesota (grouplens.org). It is used sometimes in order to study how members use MovieLens for learning how to build better recommendation systems. This dataset contain ten millions marks and 100,000 tags for 10,681 movies given by 71,567 users. Marks are based on an integer scale of 1 to 5.

We also recovered the database of IMDB (Internet Movie DataBase) which allows us to get all available information about one movie. We use it in order to add information's about the movie. We used criteria defined by these databases.

To achieve the test, we use a K-Cross fold validation [14]. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

In order to compare our results we look for a metric. We do not make prediction, so we cannot use metrics like MAE.

When items are tied in the ranking (got the same score or rate) it means that the user is indifferent between the two items. Thus, the system should not rank one item higher than the other. In such cases, rank correlation measures such as Kendall's τ can be used.

In statistics, the Kendall rank correlation coefficient, commonly referred to as Kendall's τ coefficient, is a statistic used to measure the association between two measured quantities. A τ test is a non-parametric hypothesis test which uses the coefficient to test for statistical dependence. Its value varies between -1 and $+1$, where 1 corresponds to a perfect ranking of item, -1 corresponds to the inverse order, and 0 to no correlation. In a general manner, if the value is higher than 0.4 we can say that we got a good ranking.

We use the Kendall's τ_B measure which takes into account tied values.

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_1)}}$$

With:

$n_0 = \frac{n(n-1)}{2}$
$n_1 = \sum_i t_i(t_i - 1)/2$
$n_2 = \sum_j u_j(u_j - 1)/2$
t_i =Number of tied values in the ith group of ties for the first quantity
u_j =Number of tied values in the jth group of ties for the second quantity

Moreover, we use the associated p-value, which indicates if quantities are statistically independent, that is if p-value is very low, we can assume that data are not independent.

In order to test our system, we take a sample of 100 people and we launch our two algorithms. We take into account only the marks for the following criteria: title,

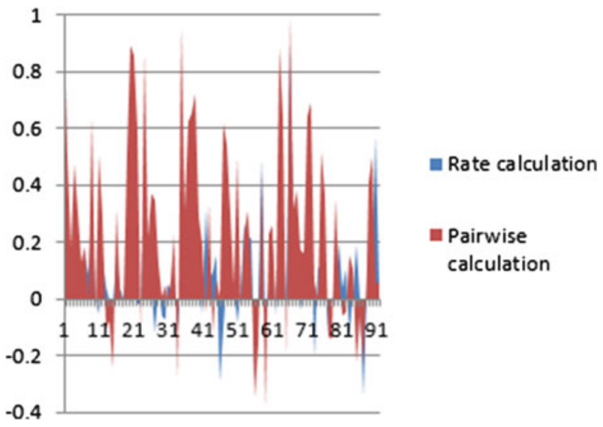

Fig. 5.2 Rate computation and pairwise computation Kendall's τ_B value for 100 users

year of the movie distribution, many styles (no limit), duration. We do no limit the number of marked movies. With the K-Cross fold method, K was equal to 6. We use an average of computed Kendall's Tau-b of each part.

We can deduce from Fig. 5.2, than our Pairwise algorithm (average 0.443 of Kendall's τ_B value) is far better than the Rate computation (average of 0.09). Moreover we can say that our system made quite good recommendation for the Pairwise algorithm because the average is higher than 0.4. This result can be explained by the ability of our system to very quickly correct erroneous weight criteria. Then, we can assume that the impact of the correction depends of the error level.

In order to produce more results, we wanted to see if our system is efficient with a large amount data. We launch our Pairwise computation on the same set of data, but we change the amount of criteria taken into account.

For this test, the used marks criteria were all the available data, up to 21 different criteria. The number is considered users stays the same: 100 users (Fig. 5.3).

We got an average of 0.233 for all available criteria taken into account. From this results we can deduce than, having more criteria didn't help. Worse, the results are poor in comparison with the situation when we use only few criteria (0.443 of Kendall's τ_B value). We can explain this bad result by the fact than our system considers more data as more noise than having more useful information. This result constitutes on the perspective that we want to work on.

Finally, a comparison between our system and some other recommender systems is made. In The results from [13] are used for this purpose.

The evaluation method uses a correlation coefficient of Spearman Rho. This factor tends to be very similar to Kendall's τ_B in practice [11]. It allowed us to directly compare the values of the coefficient of Kendall's τ_B obtained with our system (Table 5.1).

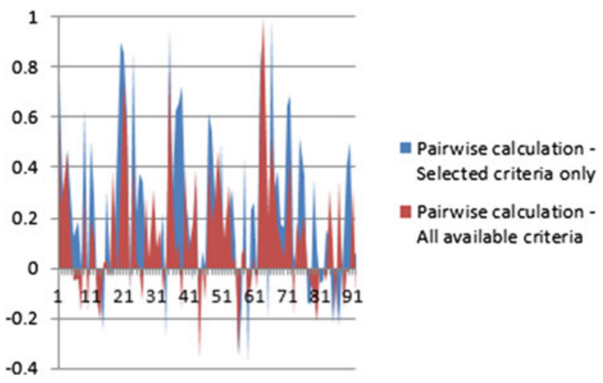

Fig. 5.3 All available criteria Kendall's τ_B value with pairwise computation for 100 users

Table 5.1 Spearman Rho coefficient for other recommender system

Method	Coefficient of Spearman Rho
Hybrid	0.3523
Content based	0.3235
Collaborative filtering	0.3214
Random	0.0104

As we can see our Rate computation (average 0.09) is very poor, but our Pairwise method (average 0.0443) gives better result than the ones got from these other recommender systems.

5.8 Conclusions and Perspectives

First of all we can assume that the implemented system satisfies the constraints and objectives which have been defined in the system requirements.

The experimentations demonstrate that our system give good recommendations, some improvements can be done. Indeed, we demonstrate that our system help users, but we do not get good performance when many criteria are used. We believe that it is possible to improve its performance in this particular case.

In order to obtain a more accurate system, we would try to use more complex computations of disaggregation and a scoring method using the Choquet integral. Indeed, it would be a major improvement for the system. For the moment, we consider that criteria are independent, but most of the time, it is not the case.

The weighted sum and, more generally, the quasi-linear medium have some weaknesses. None of these functions are capable of modeling any interaction among the attributes. Indeed, it is well known in the Utility Function theory (MAUT) that these functions lead to mutual preferential independence among the attributes, which expresses, in a sense, the independence of attributes. As these functions are not appropriate in the presence of dependent attributes, the trend has been to build the attributes supposed to be independent, which often caused errors in the assessments.

In order to obtain a flexible representation of complex phenomena of interactions among attributes or criteria (eg positive or negative synergy between certain criteria), it has proved useful to replace the weight vector by a set of non-additive functions, thus defining not only a weight for each criterion, but also on each subset of criteria [5]. Another major improvement of our system would be to introduce a set of non-additive functions as criteria.

References

1. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation, recommender systems. Papers from 1998 Workshop, Technical Report WS-98-08. AAAI Press, Menlo Park (1998)
2. Belkin, N.J., Croft, W.B.: Information retrieval and information filtering: Two sides of the same coin? Commun. ACM **35**, 29–38 (1992)
3. Billsus, D., Pazzani, M.: Learning collaborative information filters. In: Proceedings of the International Conference on Machine Learning (1998)
4. Boughanem, M., Tebri, H., Tmar, M.: Apprentissage par renforcement dans un système de filtrage adaptatif. In: CORIA, pp. 27–40 (2004)
5. Bouyssou, D., Dubois, D., Pirlot, M., Prade, H.: Concept et méthodes pour l'aide à la décision 3 - analyse multicritère, Lavoisier, Paris (2006)
6. Brans, J.-P., Mareschal, B., Vincke, Ph.: PROMETHEE: a new family of outranking methods in multicriteria analysis. In: Brans, J.-P. (ed.) Operational Research, vol. 84, pp. 408–421. Elsevier, Amsterdam (1984)
7. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference Uncertainty in Artificial Intelligence (1998)
8. Brezillon, P., Brezillon, J., Pomerol, J.Ch.: Context-based improvement of decision making: case of car driving. Int. J. Decis. Support Syst. Technol. **13**, 1–20 (2009)
9. Chen, A.Y., McLeod, D.: Collaborative filtering for information recommendation systems. In: Khosrow-Pour, M. (ed.) Encyclopedia of E-Commerce, E-Government, and Mobile Commerce, pp. 118–123. Idea Group Reference, Hershey, PA (2006)
10. Främling, K.: Modélisation et apprentissage des préférences par réseaux de neurones pour l'aide à la décision multicritère. Thèse pour L'institut National des Sciences Appliquées de Lyon (1996)
11. Fredricks, G.-A., Nelsen, R.-B.: On the relationship between Spearman's rho and Kendall's tau for pairs of continuous random variables. J. Stat. Plan. Inference **137**(7), 2143–2150 (2007)
12. Fürnkranz, J., Hüllermeier, E. (eds.): Preference Learning. Springer, Berlin (2010)
13. Garden, M., Dudek, G.: Mixed collaborative and content-based filtering with user-contributed semantic features. In: Proceedings of the 21st AAAI National Conference on Artificial Intelligence (AAAI'06), Boston, Massachusetts (2006)
14. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer, Berlin (2009)
15. Jones, Q.: SGER: Synchronous Social-Interaction-Space Recommender Systems: Core Components Model Development and Assessment. NSF IIS-HCC 0749389 (2007)
16. Maystre, L.-Y., Pictet J., Simos J.: Méthodes multicritères ELECTRE, 323 p. Presses polytechniques et universitaires romandes, Lausanne (1994)

17. Paelinck, J.: Qualiflex, a flexible multiple criteria method. Econ. Lett. **1**(3), 193–197 (1978)
18. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. **27**, 313–331 (1997)
19. Raïffa, H.: Preferences for multi-attributed alternatives, Rapport technique no. RM-58-68-DOT/RC. The Rand Corporation, Santa Monica, CA (1969)
20. Ramos, J.: Using TF-IDF to determine word relevance in document queries. In: First International Conference on Machine Learning, Rutgers University, New Brunswick NJ (2003)
21. Roy, B.: Classement et choix en présence de points de vue multiples, la méthode ELECTRE. Rev. Inf. Rech. Opér. **2**(8), 57–75 (1968)
22. Roy, B.: ELECTRE III : un algorithme de classements fondé sur une représentation floue de préférences en présence de critères multiples. Cahiers du CERO **20**(1), 3–24 (1978)
23. Roy, B.: Méthodologie multicritère d'aide à la décision. Economica, Paris (1985)
24. Roy, B., Bouyssou, D.: Aide à la dècision fondèe sur une PAMC de type ELECTRE. Document du LAMSADE, vol. 69, p. 118. Université Paris-Dauphine, Paris (1991)
25. Saaty T.: Décider face à la complexité, p. 231. Entreprise moderne d'édition, Paris (1984)
26. Schein A.I., Popescul A., Ungar L.H., Pennock D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference (2002)
27. Sutton R, Barto A.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (1998)
28. Tmar M.: Modèle auto-adaptatif de Filtrage d'Information: apprentissage incrémental du profil et de la fonction de décision. Thèse de l'Université Paul Sabatier de Toulouse, IRIT (2002)
29. Vincke P.: L'aide multicritère àă la décision, p. 179. Éditions de l'Université de Bruxelles, Bruxelles (1989)
30. Zhang Y., Callan J., Minka T.: Novelty and redundancy detection in adaptive filtering. In: Proceedings of the 25th Annual International ACM SIGIR Conference, pp. 81–88 (2002)