A dynamic approach to feature weighting

B. Arslan[1], F. Ricci[1], N. Mirzadeh[1] & A. Venturini[1]
[1] eCommerce and Tourism Research Laboratory, ITC-irst
via Sommarive 18, 38050 Povo, Italy

Abstract

The main objective of a personalized recommender system is to filter and present (recommend) to the user the most appropriate items according to his preferences. In many Case Based Recommendation systems, this goal is achieved by using weighted similarity measures. Thus, weighting the features, i.e. describing the items to be recommended, is a key issue in such systems. In this paper, we propose a dynamic weighting scheme for a Case Based Recommendation System, which is based on statistics of data extracted from past sessios. The applications of these ideas to an interactive, Case-Based travel recommender system, called Dietorecs[1], that guides European travelers for their travel decision making processes, are described.

1 Introduction

Case-Based Reasoning (CBR) systems, when confronted with a new problem, retrieve a set of similar problems from their case base, which provide a set of viable and potentially useful solutions to the new problem. In case these solutions do not fit the current situation as desired, that is if reuse of these solutions is not suitable, then they can be adapted to the new problem. The final solution, together with its original problem specification, forms a new case to be stored in the case base to "improve" future performance of the system. CBR is a cognitively appropriate approach for modeling and explaining human problem solving [9], especially in domains where experiences play an important role. Thanks to these advantages, this technique has been gaining popularity in advisory systems [4, 7].

[1]This work has been partially funded by CARITRO foundation (under contract "eCommerce e Turismo") and by the European Union's Fifth RTD Framework Programme (under contract DIETORECS IST-2000-29474).

The “problem” part of a case in a Case-Based Recommendation system is the need (or the set of needs) of the user. Therefore, one of the objectives is to capture these needs and recommend the item (or the set of items) that best meets them. So, understanding the user needs, even if they are partially described by the user, is crucial. In general, the explicit user needs, specified by the user, by means of a query language, are the main sources for the system. But, experiences show that only a few number of users express their needs properly in this way. For this reason, a vast majority of recommender systems consult the previous experiences of similar users. For instance, the collaborative-filtering approach ([11, 3]) represents a user by a vector of votes (rates and/or purchase marks) each of which corresponds to one recommendation object (product or service). Votes for those objects, that the user has not considered (rated and/or purchased) yet, are predicted by using other users’ votes, i.e. votes of users that have similar voting behaviour, where the similarity is usually computed by means of a correlation measure. In this view, the collaborative-filtering approach indexes the memory by users. Our approach is different, we use the notion of “recommendation session”, and mention each of such sessions as a “case”. With this approach, instead of similar users, we look at similar cases, i.e. previously experienced in-context similar recommendation sessions, which we call the “Reference Set”. The Reference Set is obtained by means of an on-line, dynamic computation, and is the fuel for personalized recommendations. Case components, which are hierarchically structured, are selectively exploited in the recommendation process to meet and support the user tastes.

So, one can say that we index the memory by these sessions (cases), instead of users. But, we represent the users, in a richer way, by sequences of their recommendation sessions, each of which contains extensive information (see Section 2).

The motivation of our research is a Travel Advisory System, called Dietorecs. The objective of Dietorecs system is to guide its users to build their personalized travels, i.e. aggregation (bundling) of a set of travel items. In this content, we underline the separation of two types of objects to be recommended in this domain: single travel items, and complete travel bags. A travel item is a basic travel asset like a hotel to stay, a destination to visit, an activity to perform, etc., and a travel bag is a bundle (aggregation) of such travel items. That is, a travel bag is a complete travel holding accommodation, destination, and additional activities or attractions (museum, theater, sport activities, etc.) all together. Recommending single travel items, and letting the user to build his/her own travel bag in an iterative way (e.g. first select a destination and add it to the travel bag, then a hotel, then activities, and so on) is just one facet of Dietorecs system. Dietorecs also provides complete travel recommendations, which makes it special among existing systems, none of which can support the user in building a complete travel and none of those exploit the knowledge contained in previous counseling sessions stored as cases.

Adaptation of the knowledge, gained from past experiences, to new problems, is still considered to be the most difficult step in CBR [16]. Dietorecs performs adaptation as an updating and personalization procedure, for which the Reference Set provides the required information. Dietorecs finds and recommends the up-to-date products, from its catalogues, that both satisfy the users’ needs and are similar to those chosen in similar recommendation sessions. The similarity is evaluated by means of a weighted similarity measure. In this paper, we describe our approach to this weighted-similarity recommendation process. Note that the underlined approach, is the extension of the one that has been implemented in an other Case-Based Recommendation system, such as Intelligent Travel Recommender System (ITR), which is focused on single item recommendation [13].

The rest of the paper is organized as follow: Section 2 will explain what we mean by a "case". Section 3 makes an introduction to CBR methodology. Consequently, we briefly describe the Reference Set computation in Section 4, then we will explain the underlined weighting scheme in Section 5, and finally we will conclude at Section 6 by illustrating some open issues and future works .

2 What is a Case?

We will call a complete recommendation session a case. In fact, we will use the words "case" and "recommendation session" interchangeably. So, any single user-system interaction history forms a case, which is modeled in a hierarchical tree structure. A case is decomposed into travel wish (tw), travel bag (tb), user (u), navigation history (nh) and reward (r) components. We will denote a case simply by a vector $c = (tw, tb, u, nh, r)$, but each entry in this vector is either again a vector, or a tree whose leaves are vectors as we will describe below.

- **User** (u) This block carries the user-specific profile (basically socio demographic) data for registered users, which is represented simply by a vector, e.g. $u =$ (name= John, age = 42, nationality = UK, . . .).
- **Travel Wish** (tw) This carries the user's constraints and needs about: the whole travel in general and the items specifically. That is, travel wish is consisting of feature-value pairs that are specified by the user during his interaction with the system. For the simplicity, we will assume here that we have tree types of items. For instance, these could be accommodation, destination and activity and mention them as type 1, type 2 and type 3 respectively. Mathematically, $tw = (gw, Seq(iw_1), Seq(iw_2), Seq(iw_3))$, where gw is a vector that describes the user's general wishes over the whole travel, e.g.

$$gw = [\text{budget}{<}1000,\ \text{party = couple, duration = a week}\ ];$$

$Seq(iw_j) = [iw_j^1, iw_j^2, \ldots] \quad (1 \le j \le 3)$ is the sequence of his item specific wishes for the item of type j. The motivation of using sequences for item wishes here is that the user may have two or more items of the same type in his travel bag, and for each one he can provide different wishes. For example, $Seq(iw_3) = [[iw_3^1, iw_3^2, iw_3^3]$ and

$$iw_3^1 = [\text{type = ski, location = Cavalese, altitude} > 800\ ],$$
$$iw_3^2 = [\text{type = biking, location = Trento, difficulty level = advance}\ ],$$
$$iw_3^3 = [\text{type = fishing, location = Trento, lake = true}]$$

- **TravelBag** (tb) Travel Bag has a complex data structure that collects together the items selected during a case session. It is represented in a tree-like hierarchical way, denoted $tb = (gd, d_1, d_2, d_3)$, where gd is a vector that describes general properties of the travel bag (e.g , gd =(1100, couple, hotel, destination, activity), and $d_i = [d_{i,1}, d_{i,2}, \ldots],\ (1 \leq i \leq 3)$ describes the item(s) of type i in the bag, $d_{i,j}$ stands for the description of the j^{th} item of type i in the travel bag.
- **Navigation History** (nh) This block stands for tracking the user's clicking behavior. Shortly we can say that *nh* is more than standard user logs. We model each session stage (i.e. status-quo between two function calls) as a triple: (F,I,O), where F is the function called (e.g. search, find similar item, etc.), I is the input of the function (e.g. wishes specified by the user, description of an item, etc.). Basically, nh is a sequence of such triples.
- **Reward** (r) Reward is a collection of rates (tree-structured like the travel bag) provided by the user for each item in the travel bag. More precisely $r = (gr, [r_{11}, r_{12}, \ldots], [r_{21}, r_{22}, \ldots], \ldots)$, where r_{11} is the rate for the first item of type 1, r_{12} is the rate for the second item of type 1, r_{21} is the rate for the first item of type 2, etc. gr is the general rate for the whole travel bag computed as a composite average of rates of items.

3 CBR cycle

The CRB problem solving methodology is usually depicted as a learning loop that comprises 5 basic steps ([1]):

- 1. *Retrieve:* Given a problem, retrieve a set of stored cases (e.g. problem-solution couples).
- 2. *Reuse:* Apply one or more solutions from these retrieved cases, perhaps by combining them with each other or with other knowledge sources.
- 3. *Revise:* Adapt the retrieved solution(s), as needed, in an attempt to solve the new problem.
- 4. *Review:* Evaluate the outcome(s) when applying the constructed solution to the current problem. If the outcome is not acceptable, then the solution will require further revision.
- 5. *Retain:* Consider adding the new learned case (a new problem + solution couple) to the case base[2]

We follow the same methodology, but with some remarkable distinctions from the classical picture. Details of Dietorecs' CBR cycle can be seen in [2], but here we want to underline that in addition to the memory of experiences (i.e. the case base), we also exploit the catalogues, which hold up-to-date items. In Dietorecs, the case base contains the structured cases, i.e. hierarchically structured past recommendation sessions as we have described in Section 2. The travel items selected

[2]The author in [1] uses "<problem, solution, outcome> triples" instead of problem-solution couples, and "Case Library" instead of Case Base. But, for the sake of consistency with the rest of the paper and to avoid any ambiguity we will follow the postulated terminology above.

in a recommendation session, i.e. the items that form the travel bag, are actually pointers to items in catalogues. The case base provides good candidate products to be recommended and the information about aggregations of such products. We use the case base for learning such knowledge and for ranking items selected in the catalogues (see Figure 1). The catalogues are exploited for obtaining up-to-date information about currently available services. So, instead of reusing exactly the same recommendations in past, we always recommend up-to-date items to the user. Simply, having the Reference Set (and hence good candidate items to be recommended to the user), we go to the catalogues and select up-to-date items that are similar to those in the Reference Set (as shown in Figure 1), if necessary re-bundle them, and deliver to the user already revised (updated) recommendations.

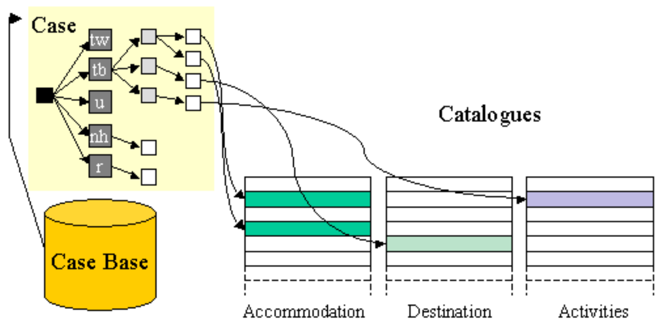

Figure 1: Case Base and Catalogues

4 Reference Set

As we have said elsewhere, the dynamic computation of the Reference Set requires the comparison of two non-trivial structures, i.e. similarities between hierarchically structured cases. If we denote the active case (the current recommendation session in which the travel bag is under construction) by $c = (tw, tb, u, nh, r)$, we compute the similarity between c and an other case $c' = (tw', tb', u', nh', r')$ from the case base as follows:

$$\begin{aligned} Sim(c,c') = \frac{1}{5}(&w_w Sim(tw,tw') + w_b Sim(tb,tb') \\ &+ w_{wb} Sim(tw,tb') + w_u Sim(u,u') + w_r Rew(r')) \end{aligned} \tag{1}$$

where $w_w, w_b, w_{wb}, w_u, w_r$ (real numbers between 0 and 1) are the weights that are used to balance the relative components importance in case similarity computation[3]

Note that the "cross" similarity $Sim(tw, tb')$ between the travel wishes of c and the travel bag of c' is typically used when a new case c is going to be built. At that point the system must match the wishes of c with the bag of c', and not only the user wishes that were at the base of c' (i.e. $Sim(tw, tw')$). First four components stand for the matching process, that is, this is how we match the problem description of the active case c with that of c'. Moreover, the term $Rew(r')$ does not represent any comparison between cases, but takes into account the importance of the retrieved case. The motivation here is to consider primarily those highly rated (i.e. successful) travel bags, so having a better rate makes that case more important. As a consequence, Equation 1 makes more similar a case that has been rated highly by the user that built the case. Thus, the similarity in Equation 1 copes with the components of cases in a quite assiduous way and yields a careful Reference Set.

For each of the similarities in Equation 1, a different similarity measure needs to be considered. Detailed descriptions of similarity computations can be seen in [12], but here we want to mention that even the sub-similarity components in Equation 1 (e.g. $Sim(tw, tw')$) are complex comparisons. Because, tw and tb (and even r) components are tree-like hierarchical structures.

5 Similarity Based Recommendation

In this section, we will focus on weighted similarity computations that we perform (on-line) in order to get the up-to-date items from the catalogues. Namely, we refer to "scoring and ranking" issues. The idea is to score the catalogue items with respect to the user's wishes and deliver to him a ranked list of recommendations. But, additional to those wishes explicitly described by the user, we also exploit the Reference Set.

For the simplicity, the idea is illustrated in a scenario where the user u of the active case $c_u = (tw, tb, u, nh, r)$ is looking for an accommodation item to be added to his travel bag tb. Using c_u as a probe, the system forms the Reference Set by retrieving the cases from the case base that are similar to c_u (as explained in Section 4). Let us denote the obtained Reference Set by $\bar{R}$, and assume that, currently, the user's accommodation specific wishes are as follows:

$$iw_1^1 = \{ \text{ location = Alto Adige, price } < 1000, \text{ aircondition = true } \}$$

These explicit wishes are translated by the system to a machine understandable query Q_u, and the result set of the query Q_u defines the set of catalogue items

[3]Equation 1 is generated with the spirit of the principle "two objects are similar if they contain similar objects" [8], which is a quite customized version of the integration functions proposed in [8], and other CBR approaches [10, 6].

that is going to be considered for the recommendation process[4]. The idea is that an item is to be recommended if the following two general conditions are satisfied:

- The item satisfies the constraints (wishes) that are explicitly specified by the user (i.e. the query constraints in Q_u).
- The item must be similar to those items selected by users in "similar" recommendation sessions (i.e. the items that appear in the Reference Set).

The accommodation solutions, in the Reference Set, that the users have picked (and added to their travel bag) in these sessions, are good candidates to be recommended to the current user. With this belief, we compute the similarity between the accommodation items in the Reference Set and those in the catalogues. Items are represented by simple heterogeneous vectors, i.e. vectors consisting of both numeric and symbolic valued components. Therefore, if $x_{\bar{R}} = (x_1, x_2, \ldots, x_n)$ is an accommodation item that appears in the Reference Set, and $y = (y_1, y_2, \ldots, y_n)$ is an accommodation item from the catalogue that satisfies the user's constraints (i.e. y is in the result set of the query Q_u), then we compute the similarity

$$Sim(x_{\bar{R}}, y) = e^{-\left(\frac{1}{\sum_{i=1}^{n} w_i}\sqrt{\sum_{i=1}^{n} w_i (d_i(x_i, y_i))^2}\right)} \tag{2}$$

where $d_i(\cdot,\cdot)$, $(1 \le i \le n)$ is a customized version of the Heterogeneous Euclidean Overlap Metric (see [15]), defined as follows:

$$d_i(x_i, y_i) = \begin{cases} 1 & \text{if the } i\text{-th feature is symbolic and } x_i \neq y_i \\ 0 & \text{if the } i\text{-th feature is symbolic and } x_i = y_i \\ \dfrac{|x_i - y_i|}{4\sigma_i} & \text{if the } i\text{-th feature is numeric} \end{cases} . \tag{3}$$

Note that, if the i-th feature is numeric we divide the Euclidean distance $|x_i - y_i|$ by four standard deviations $4\sigma_i$ to normalize the distance and exclude the outlying values.

Now, we are trying to understand what features we should consider more importantly while computing this similarity. In other words, what are the weights $\{w_i\}$ in the Equation 2.

Let F_{c_u} be the set of features explicitly specified by the user of the active case (in this example we have $F_{c_u} = \{$ location, price, air-condition $\}$). Like in F_{c_u}, for each of the cases in $\bar{R}$ we have a list of features that were specified during their interactions with the system. So, for all cases $c \in \bar{R}$ we have $F_c = \{f_{c_1}, f_{c_2}, \ldots f_{c_{k_c}}\} \quad (c_{k_c} \le n)$. If we denote the set of all different features that appeared in the Reference Set by $F_{\bar{R}}$, i.e. $F_{\bar{R}} = \cup_{c \in \bar{R}} F_c$, we will consider the union $F = F_{c_u} \cup F_{\bar{R}}$.

[4]For the case of failure in the query results (too many or no result set), Dietorecs performs an interactive/intelligent query management task. Because of the lack of the space we are not going into details of these issues, but interested readers may find more information in [13].

For the simplicity, let us assume that what we obtained in this example is as follows:

$$\begin{aligned} F &= F_{c_u} \cup F_{\bar{R}} = \{f_1, f_2, \cdots, f_7\} \\ &= \{\underbrace{\text{location, price, air-condition}}_{F_{c_u}}, \text{ TV, catering, sofa-bed, mini-bar } \} \end{aligned}$$

Obviously, among all the features in F, those that are explicitly specified by the user (i.e. the features in F_{c_u}) are more important. Therefore, we give the highest weights to them. Namely, if the values of weights $\{w_i\}$ vary in the unit interval (i.e. $w_i \in [0, 1]$ for all i), then we assign $w_1 = w_2 = w_3 = 1$. Now we want to assign the weights for the features in $F \setminus F_{c_u} = \{f_4, f_5, f_6, f_7\}$, for which we will use their frequencies and those with higher frequencies will be weighted higher. Let us assume that the cardinality $|\bar{R}|$ of the Reference set is 6 (there are 6 recommendation sessions found similar to c_u) and consider the features in $F \setminus F_{c_u}$ and their frequencies (occurances in the set $\bar{R}$) to be as in the left part of the Table 1. For instance, the frequency of the feature $f_4 =$ TV is 4, which means that only in 4 of the recommendation sessions in $\bar{R}$ the users have wished to have a TV equipment in their accommodations.

Proportional to their frequencies, the values of the weights are assigned in a simple way, e.g.

$$w_i = \frac{\text{frequency of the feature } f_i}{|\bar{R}|}$$

So, in this example the weighting scheme will be as shown in Table 1.

	feature	frequency	weight
f_1:	location	-	$w_1 = 1.00$
f_2:	price	-	$w_2 = 1.00$
f_3:	air-condition	-	$w_3 = 1.00$
f_4:	TV	4	$w_4 = 0.66$
f_5:	catering	5	$w_5 = 0.83$
f_6:	sofa-bed	1	$w_6 = 0.16$
f_7:	mini-bar	2	$w_7 = 0.33$

Table 1: Features, frequencies and weights

In this case, for the scoring and ranking, the distance between an accommodation item $x_{\bar{R}}$ from the Reference Set $\bar{R}$ and an accommodation item y from the catalogue will be measured as follows:

$$Sim(x_{\bar{R}}, y) = e^{-\left(\frac{1}{4.98}\sqrt{D(x_{\bar{R}}, y)}\right)}$$

where

$$\begin{aligned} D(x_{\bar{R}}, y) = & d_1^2(x_1, y_1) + d_2^2(x_2, y_2) + d_3^2(x_3, y_3) + 0.66\, d_4^2(x_4, y_4) \\ & + 0.83\, d_5^2(x_5, y_5) + 0.16\, d_6^2(x_6, y_6) + 0.33 d_7^2(x_7, y_7) \end{aligned}$$

6 Conclusions and Future Work

Dietorecs has been partially implemented and is still under development. Together with its other parts the underlined weighting scheme will be validated in the evaluation phase.

Similar statistical approaches for feature weighting have already been adopted by others. For instance, TripMatcher (see for example [5]) predicts the user's explicit interests to features that has not been selected by the user. But, for which, among all such features, it picks the ones that have been most frequently selected by similar users. TripMatcher uses positive and negative evidence of the user's interests, where the explicit user rates and expert human evaluations over recommendation objects are the main sources. In many systems, users must provide explicit ratings to express their attitudes, but, this requires additional user effort and hence is not desirable by the users. Another example is the Web-based ELFI information system (e.g. [14]), that uses a univariate significance analysis to determine if a user is interested in specific values of the features. It is based on the assumption that attribute values in random samples are normally distributed. If the considered value appears in relevant sessions more frequently than in a random sample, the user is supposed to be interested in that value of the feature.

Among these two, one can say that our approach is closer to that of ELFI system. But, there are slight differences. They customize the classical instance based classification approach where the assumption is that recommendation objects can be classified as "interesting" or "not interesting" for each of the user profiles. We are postulating a similarity-based recommendation approach rather than classification-based ones.

Acknowledgement: Authors are thankful to Antonella Spada for her comments and suggestions.

References

[1] D. W. Aha. The omnipresence of case-based reasoning in science and application. *Knowledge-Based Systems*, 11(5-6):261–273, 1998.

[2] B. Arslan and F. Ricci. Case based session modeling and personalization in a travel advisory system. In F. Ricci and B. Smyth, editors, *Proceedings of*

the AH'20002 Workshop on Recommendation and Personalization in eCommerce, pages 60–69, Malaga, Spain, May, 28th 2002.

[3] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998. Morgan Kaufmann Publisher.

[4] R. Burke. *Encyclopedia of Library and Information Science*, volume 69, chapter Knowledge-based Recommender Systems. Marcel Dekker, 2000.

[5] J. Delgado and R. Davidson. Knowledge bases and user profiling in travel and hospitality recommender systems. In *Proceedings of the ENTER 2002 Conference*, pages 1–16, Innsbruck, Austria, January 22-25 2002. Springer Verlag.

[6] M. Doyle and P. Cunningham. A dynamic approach to reducing dialog in on-line decision guides. In *Advances in case-based reasoning: 5th European workshop, EWCBR-2000, Trento, Italy, September 6–9, 2000: proceedings*. Springer, 2000.

[7] M. H. Göker and C. A. Thomson. Personalized conversational case-based recommendation. In *Advances in case-based reasoning: 5th European workshop, EWCBR-2000, Trento, Italy, September 6–9, 2000: proceedings*, pages 99–111. Springer, 2000.

[8] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. Technical report, Stanford University, October 2001.

[9] D. B. Leake, A. Kinley, and D. C. Wilson. Case-based similarity assessment: Estimating adaptability from experience. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA*, pages 674–679, 1997.

[10] E. Plaza. Cases as terms: a feature term approach to the structured representation of cases. In *International Conference on Case-Based Reasoning (ICCBR-95), Sesimbra, Portugal, Oct. 23-26*. Springer Verlag, 1995.

[11] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[12] F. Ricci, B. Arslan, N. Mirzadeh, and A. Venturini. Cbr methodologies in dietorecs. Technical Report D4.1, DieToRecs IST-2000-29474, 2002.

[13] F. Ricci, D. Blaas, N. Mirzadeh, A. Venturini, and H. Werthner. Intelligent query management for travel products selection. In *ENTER 2002 Conference*, Innsbruck, Austria, January 22-25 2002.

[14] I. Schwab, W. Pohl, and I. Koychev. Learning to recommend from positive evidence. In *Proceedings of Intelligent User Interfaces 2000*, pages 241–247, ACM Press, 2000.

[15] D. Wilson and T. Martinez. Improved heterogenous distance functions. *Journal of Artificial Intelligence Research*, 11(6):1–34, 1997.

[16] R. B. Wolfgang Wilke. Techniques and knowledge used for adaptation during case-based problem solving. In *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems,IEA98*, Spain, 1998.