

# Product Recommendation with Interactive Query Management and Twofold Similarity

Francesco Ricci, Adriano Venturini, Dario Cavada, Nader Mirzadeh, Dennis Blaas, and Marisa Nones

eCommerce and Tourism Research Laboratory  
ITC-irst  
via Sommarive 18  
38050 Povo, Italy  
{ricci,venturi,dacavada,mirzadeh,blaas,nones}@itc.it

**Abstract.** This paper describes an approach to product recommendation that combines in a novel way content- and collaborative-based filtering techniques. The system helps the user to specify a query that filters out unwanted products in electronic catalogues (content-based). Moreover, if the query produces too many or no results, the system suggests useful query changes that save the gist of the original request. This process goes on iteratively till a reasonable number of products is selected. Then, the selected products are ranked exploiting a case base of recommendation sessions (collaborative-based). Among the user selected items the system ranks higher items that are similar to those selected by other users in similar sessions (twofold similarity). The approach has been applied to a web travel application and it has been evaluated with real users. The proposed approach: a) reduces dramatically the number of user queries, b) reduces the number of browsed products and c) the selected items are found first on the ranked list.

## 1 Introduction

Recommender systems are applications exploited in e-commerce sites to suggest interesting and useful products and provide consumers information to facilitate the decision-making process [12]. Recommender systems research is motivated primarily by the need to cope with: information overload, lack of user knowledge in a specific domain, cost-benefit tradeoff optimization, interaction cost minimization.

Building real operational recommender systems that tackle all the above problems is extremely complex and requires: long user requirements elicitation, task modelling, tuning of recommendation algorithms, development and test of the graphical user interface. This holistic approach has been stressed in our previous paper [10] where we described an application to trip planning, aimed at recommending to a leisure traveller a good bundling of trip products (locations, accommodation, attractions, activities, events). In this paper we want to focus on the basic recommendation technology that we have developed, stressing the

general aspects, its wider applicability, and presenting the empirical evaluation results conducted with real users. The ultimate goal of the proposed recommendation technology is to help its users through a ranked list of products (from electronic catalogues) taking into account a wide variety of needs and wants (explicit vs. implicit, must vs. optional, long term vs. short term, personal vs. group) in an efficient and effective way.

Recommender systems implicitly assume that user's needs and constraints can be converted/mapped, by means of appropriate recommendation algorithms, into product selections using "knowledge" managed by the system. Burke distinguishes three types of recommendation approaches: collaborative- or social-filtering; content-based and knowledge-based [4]. Here we distinguish only between collaborative- and content-based assuming a more comprehensive definition of knowledge. Hence, for instance, even a simple archive of sequences containing the items bought by users, is a source of knowledge for a collaborative-based approach <sup>1</sup>.

In content-based filtering approaches, the user expresses needs, benefits and constraints and the system matches this description with items contained in a catalogue of products. Products are typically ranked according to the degree of matching with the user query. Content-based approaches may possibly exploit the history of past user queries to build a user profile, ultimately stressing the importance of matching the needs collected during a specific recommendation session. Case-Based Reasoning recommendations have often been viewed as a derivation of content-based approaches, where the problem description comprises the user needs and constraints and the solution(s) is (are) the items retrieved. Content-based approaches appear to have as a major drawback to be unable to produce a sufficient degree of diversity, as they stick to the explicit needs expressed in the user query [15]. Moreover, designing dialog support techniques aimed at helping the user to express explicitly his needs has always been considered as a major issue in the CBR community. This has motivated a lot of research on conversational case-based reasoning [8, 9], non-standard retrieval approaches (order-based [3], query tweaking [5]), attribute selection methods [13], structured case retrieval [6].

Collaborative-based approaches [2] collect user ratings on offered products and/or user previously bought product information to identify similarities between users and cross recommend to a user those products, not yet considered, and highly ranked or bought by similar users. Collaborative approaches are strong in suggesting new items that the user may not know. Conversely, collaborative-based approaches require a vast amount of user feedback before producing satisfactory recommendations and are not able to take into account session-dependent (contextual) user needs. Moreover collaborative filtering is applicable only when products to be recommended are standardized, i.e. sold in the same form to several users, and a user is likely to buy many items of the same

---

<sup>1</sup> Actually there are other recommendation methodologies, see for instance [5], but because of lack of space we limit the discussion to collaborative- and content-based approaches that are by far the most popular.

type through the same web site. Product types that do have these characteristics include CDs, books and movies. But there are other products (e.g. travels, cameras, cars, computers) that are not amenable to collaborative-based recommendation. It is not likely that information about the cars previously owned by a user could predict the next car a user may buy. Or, the fact that a user once travelled to New York does not mean that the user will never go there again (a collaborative-based recommender does not suggest twice the same product).

The above mentioned shortcomings of both content- and collaborative-based filtering has motivated us to design a more powerful methodology, which is described in Section 2. The formal definition of case base and query model is given in Section 3. Section 4 describes the whole recommendation process and its sub-components: the query evaluation and product rankings algorithms. The empirical evaluation of the proposed methodology is given in Section 5. Finally, Section 6 gives the conclusion and outlines some future work.

## 2 A New Methodology for Recommender Systems

We have designed a novel hybrid collaborative/content based recommendation methodology that is further motivated by the following requirements:

- Products may have a complex structure, where the final recommended item is an aggregation of more elementary components. For instance a trip may bundle a flight, an accommodation and a rental car, or a desktop computer may be sold together with a printer, a monitor and a scanner.
- Both short term (goal oriented) preferences and long term (stable) preferences must influence the recommendation. Short term preferences are highly situation dependent, tend to be hard constraints, and should dominate long term preferences. For instance, if the user searches for a business flight the system must shade the influence of a previous history of 'no frills' flights bought by the user for a leisure travel.
- The recommendation methodology must fit into the ubiquitous form-based information search interfaces that the majority of e-Commerce web systems provide. This would make the methodology seamlessly pluggable into existing systems.
- The lack of an initial memory of any user interactions with the system or buying should not prevent completely the applicability of the methodology. Thus, unregistered users should be allowed to get recommendations.

In the proposed methodology a case models a unique human-machine interaction session and collects: all the information provided by the user during the session, the products selected, and some stable user related preferences and demographic data if he is registered. A recommender system based on this methodology stores these cases in a repository (case base) in addition to the catalogues of products (databases). This is in contrast with standard CBR recommender systems that view the catalogue of products as a case base.

All input features provided by the user during an interaction session fall into two (not mutually exclusive) categories: content and collaborative features. Content features are those features that can be constrained in the users' queries, and they are used as descriptors of the products in the catalogues. For instance, in a hotel description the hotel category and the presence of a garage are content features. Conversely, the nationality of the user or the travel purpose could be used to describe a trip, and naturally is not part of the any description of the products found in the catalogue. In this setting these are collaborative features since they can be used to measure user/session similarity.

When the recommender searches for similar recommendation sessions, it uses the collaborative features acquired from the user. We have applied various strategies to obtain these features. In our first recommender system (ITR [10]), they are collected in the first stage of the interaction, whereas a second application (Dietorecs [7]) even the products selected at a given point play the role of collaborative features. After a small set of similar recommendation sessions is computed, the items in the result set of the user query are ranked using a double similarity computation. A score for each item in the result set is computed by maximizing the product of the case similarity (between the current case and the retrieved cases) and the item similarity (between a selected item and the item of the same type contained in the retrieved case). The selected items are finally presented to the user in decreasing score order. The overall effect is that content features dominates the selection process and determine what is in the result set. Then the collaborative features are exploited to order the result set, popping up items that are recommendable because they are similar to those selected by other users in similar recommendation sessions.

We have empirically evaluated the proposed methodology in a between subjects test. A first users group planned a vacation with a system built with the proposed recommendation methodology. A second group solved the same task with a system variant in which we discarded the interactive query management and ranking functions. We have proved that the proposed methodology drastically reduces the number of queries issued, the number of items that the user browsed, and sorts the result list in such a way that selected items are displayed earlier.

### 3 Case Model

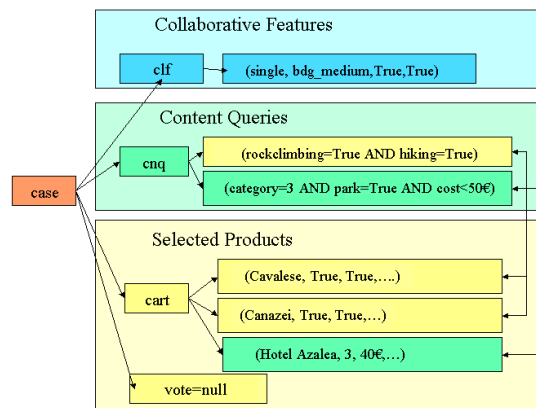
In our approach, a case represents a user interaction with the system, and therefore is built incrementally during the recommendation session. A case comprises the following main components:

- **Collaborative Features (clf)** are features that describe general user's characteristics, wishes, constraints or goals (e.g. desire to relax or to practise sports). They capture preferences relevant to the user's decision-making process, which cannot generally be mapped into the features of products in the electronic catalogues. These features are used to measure case (session)

similarity. A knowledge of the domain and the decision process is essential to select the right collaborative features [10].

- **Content Queries (cnq)** are queries posed over the catalogues of products. Content queries are built by constraining (content) features that describe products listed in the catalogues. Products may belong to different types (e.g. an accommodation or an event).
- **Cart** contains the set of products chosen by the user during the recommendation session represented by the case. A cart represents a meaningful (from the user's point of view) bundling of different products. For instance, a travel cart may contain some destinations, some accommodations, and some additional attractions.
- **Vote** is a collection of votes given by the user to the products contained in the cart.

Figure 1 shows an example in the tourism domain. It represents a user, who is single, has a medium budget, and is looking for a vacation (destinations and accommodation) where he can practice some sports and relax (collaborative features). Then there are the queries (over the product catalogues) constructed by constraining content features of the products: for instance, he wants to stay in a three star hotel, which has a private parking lot and has a cost per night less than 50 Euro. The destination should be a resort suitable for rock climbing and hiking activities. Given these preferences, the user is supposed to have selected and added to his cart the Azalea Hotel, and the Cavalese and Canazei resorts. In this example, the user has selected two destinations by querying the destination catalogue. Note that, the user has not yet specified any vote over the selected products (vote=null).



**Fig. 1.** An example of case. The collaborative features are TravelParty, Budget, Sports, and Relax.

More formally, the Case Base (CB) is defined as following:

$$CB \subseteq CLF \times CNQ \times CART \times V$$

*Collaborative Features Space (CLF)* is a vector space  $CLF = \prod_{i=1}^l CLF_i$ , where  $CLF_i$  is set of symbols, a finite subset of the integers or a real interval. In our example,  $CLF = TravelParty \times Budget \times Sports \times Relax$ . Where  $TravelParty = \{single, family\}$ ,  $Budget = \{low, medium, high\}$  and  $Sports = Relax = \{True, False\}$ . In the example shown, the features specified by the users are  $clf = (single, medium, True, True)$ .

*Content Queries (CNQ)* is the space of all the queries that the user can specify over products in the catalogues. We assume that each product  $p$  can be represented as a vector of features  $X = \prod_{i=1}^n F_i$ .  $F_i$  can be, as above, a set of symbols, a finite subset of the integers or a real interval. A catalogue  $CX \subset X$  is said to be of type  $X$ . A query  $q$  over a catalogue  $CX$  is the conjunction of simple constraints, where each constraint involves only one feature. More formally,  $q = (c_1, \dots, c_m)$ , where  $m \leq n$ , and:

$$c_k = \begin{cases} f_{i_k} = v_k & \text{if } F_{i_k} \text{ is symbolic} \\ l_k \leq f_{i_k} \leq u_k & \text{if } F_{i_k} \text{ is finite integer or real} \end{cases} \quad (1)$$

where  $f_{i_k}$  is a variable of type (with domain)  $F_{i_k}$ ,  $v_k \in F_{i_k}$ , and  $l_k, u_k \in F_{i_k}$  are the boundaries for the range constraint on a real valued feature. Let  $Q(X)$  be the space of all the queries over  $X$ . Furthermore, let us assume that there are  $N$  product types  $X_1, \dots, X_N$ . Then we denote by  $Q = \bigcup_{i=1}^N Q(X_i)$  the space of all the queries on the catalogues. We finally denote with  $CNQ = \mathcal{P}(Q)$ , the set of all subsets of queries over  $X_i, \dots, X_N$ , i.e.

$$CNQ = \{cnq = \{q_1, \dots, q_k\} \text{ s.t. } q_i \in Q(X_{j_i})\}$$

In the example shown in Figure 1,  $cnq = \{(rockclimbing = true \text{ AND } hiking = true), (category = 3 \text{ AND } park = true \text{ AND } cost \leq 50)\}$ .

*CART* is defined as  $CART = \mathcal{P}(\bigcup_{i=1}^N X_i)$ , i.e. an element  $cart \in CART$  is subsets of products:  $cart = \{p_1, \dots, p_k\}$  such that  $p_i \in X_{j_i}$ .

*Vote (V)* represents the case evaluation expressed by the user who has built it. The vote in fact is structured similarly to the cart, i.e., there is a elementary vote for the cart as a whole and elementary votes for the items in the cart. An elementary vote is an integer between 0 and 5. In this paper we shall not deal with this component since collaborative-filtering in our approach is not based on user votes (see [1] for a description of its usage in a more general case similarity metric).

## 4 Product Recommendation

This section describes the proposed approach to support the user in the selection and recommendation of products. The overall process is shown in Figure 2. The user interacts with the recommender system by querying recommendations about a product type (e.g., a destination). To simplify the notation, we will consider just one product space  $X$ , and  $q \in cnq$  will denote the user’s query. The system either recommends some products, or, in case the query fails, suggests some query refinements. The RecEngine module manages the request. First, it invokes the EvaluateQuery function (1) of the Intelligent Query Manager module (IQM), by passing the query. This function searches the catalogue for products matching the query. If too many<sup>2</sup> or no product matches the input query  $q$ , then EvaluateQuery analyzes  $q$  and determines a set of query refinements to suggest. These refinements are presented to the user as a set of features. If there are too many results then three features are selected and the user is asked to provide a value for one of them to narrow down the search result (attribute selection). Conversely if no result can be found the system explains to the user the cause of the failure, i.e., it lists those constraints that if relaxed would allow the query to return some results. The EvaluateQuery function is described in Section 4.1.

When the number of items retrieved is satisfactory then the products are ranked by invoking the Rank method (2). Rank receives the collaborative features  $clf$  of the current case and the set of products selected by the EvaluateQuery function. The  $clf$  features are used to retrieve the 10 most similar cases, and the products contained in these retrieved cases are then used to rank the user selected products. Finally, the ranked products are returned to the user. The Rank algorithm is described in Section 4.2.

### 4.1 EvaluateQuery Algorithm

The goal of the EvaluateQuery algorithm is to find out the products matching the content query expressed by the user. EvaluateQuery and the Intelligent Query Management module are aimed at helping the user to reformulate the query when failures occur. Because of lack of space we only sketch the description of that module. The reader is referred to [11] for further details.

The EvaluateQuery algorithm is described in Figure 3. It receives as input a query  $q$ , over a product catalog  $CX$ . It returns the products  $P$  matching the query  $q$ , and a set of query refinement suggestions  $R$ .  $R$  is a set of triples, each containing a feature  $f_{i_k}$ , a constraint  $c_k$  over the feature  $f_{i_k}$ , and the suggested operation  $op_k$  (add, modify, or remove from  $q$ ). In line 1, the SearchCatalog function is invoked, passing the  $q$  query as parameter. The function searches through the catalogue for products matching  $q$ , and returns the set of matching products. Line 2 evaluates the size of the result set. If the number of selected products is above a certain threshold, the TightenQuery function is invoked (line 3). This function, using information related to the product catalogue data

---

<sup>2</sup> in our experiments this threshold was set to 10.

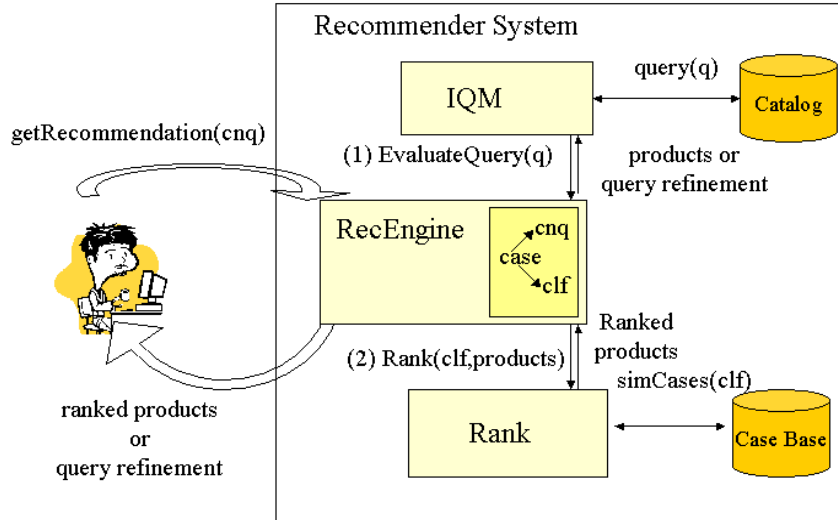


Fig. 2. Recommendation process.

distribution (entropy and mutual information), returns a set of features (three) suggested to the user to further constrain the search. The user can chose one (or more) and provide a value (symbolic feature) or a range of values (numeric feature).

Actually, TightenQuery returns a set of triples  $(f_i, null, add)$ , where  $f_i$  is a feature and  $c_i = null$ , since the TightenQuery function cannot guess the exact value (or range of values) the user may want to specify for a suggested feature. Line 5 tests the empty result set condition. If the result set is empty, the RelaxQuery function is called (line 6). This function searches for those  $q$  modifications (constraint relaxation) that will allow  $q$  to retrieve some results. The suggested modifications are again returned as set of triples  $(f_i, c_i, op_i)$ , where  $op_i = remove$  for symbolic features and  $op_i = modify$  for finite integer or real features, and  $c_i$  represents the new (larger) range to be set. If neither relaxation nor tightening is invoked, the result set  $P$  is returned (line 9).

#### 4.2 RankProducts Algorithm

The RankProducts algorithm ranks the products retrieved by EvaluateQuery. Ranking is computed exploiting two similarity metrics: first, the case base is accessed to retrieve the (10) most similar cases (reference cases) to the current one. This similarity-based retrieval uses the collaborative features. Then, the products contained in the carts of the retrieved reference cases (reference products) are used to sort the products selected by the user's query. The basic idea is that among the products in the result set of the query one will get a better score if it is similar/equal to a product (of the same type) bought by a user with similar needs and wants. The Figure 4 describes the algorithm in detail.

$CX$  is the product catalog

$q$  is the user's query

$P = \{p_1, \dots, p_k\}$  the products selected by the user query  $q$ .

$R = \{(f_{i_1}, c_1, op_1), \dots, (f_{i_m}, c_m, op_m)\}$ ,  $op_j \in \{add, modify, remove\}$ ,  $c_j$  is a constraint on feature  $f_{i_j}$  to be: added, modified or removed ( $op_j$ )

---

```
EvaluateQuery( $q, CX$ )
1  $P \leftarrow SearchCatalog(q, CX)$ 
2 if  $Size(P) > threshold$ 
3    $R \leftarrow TightenQuery(q, CX)$ 
4   return  $R$ 
5 else if  $Size(P) = 0$ 
6    $R \leftarrow RelaxQuery(q, CX)$ 
7   return  $R$ 
8 else
9   return  $P$ 
```

**Fig. 3.** The EvaluateQuery algorithm.

The Rank function receives the current case  $c$ , the set of products  $P$  retrieved by the function EvaluateQuery, and the case base  $CB$ . It returns the products  $P$  ranked. First it retrieves from  $CB$  the reference cases  $RC$  (line 1). Here, the similarity query uses the collaborative features  $clf$ . In line 3,  $RP$  is defined as the products contained in the reference cases  $RC$ <sup>3</sup>. Note that products in  $RP$  and  $P$  are of the same type. In line 4, the Score of each product  $p_i$  is computed as the maximum of  $Sim(c, rc_j) * Sim(p_i, rp_j)$  over all the reference products  $rp_j$  (the similarity functions are described in Section 4.3).

Computing the final product score as the multiplication of cases and products similarity mimics the collaborative-filtering (CF) approach, but there are some notable differences. First, differently from a standard CF approach, only the first nearest neighbor case is considered, i.e., the case that yields the maximum value for  $Sim(c, rc_j) * Sim(p_i, rp_j)$ . The rationale for this choice is that we can use the retrieved case to explain the score value to the user. Secondly, we do not consider the votes given by other users to the product to be scored, as is common in CF. Conversely, we use the similarity of the product to be scored to products selected in a similar case (user session) as a sort of implicit vote: if another user in a similar session has chosen that product or a similar one, this is an indication that this product fits the needs that are shared by the current user and the previous user.

Let us consider the following simple example. Let us assume that the collaborative features in  $CLF$  are TravelParty (symbolic), Budget (numeric), Sports

---

<sup>3</sup> For sake of simplicity we assume that each reference case  $rc_i$  contains just one product  $rp_i$  of the same type of the products to be ranked, but the same approach applies also when more products are contained in a case.

$RC = \{rc_1, \dots, rc_{10}\}$  retrieved cases  
 $RP = \{rp_1, \dots, rp_{10}\}$  products inside the reference cases  
 $c$  is the current case  
 $CB$  is the case base  
 $P = \{p_1, \dots, p_k\}$  the products selected by the user query

---

$Rank(c, p, CB)$   
 1  $RC \leftarrow FindSimilarCases(c, CB)$   
 2  $RP \leftarrow ExtractReferenceProducts(RC)$   
 3 **for each**  $p_i \in \{p_1, \dots, p_k\} = P$   
 4      $Score(p_i) \leftarrow \max_{j=1 \dots 10} \{Sim(c, rc_j) * Sim(p_i, rp_j)\}$   
 5  $P \leftarrow Sort\{p_1, \dots, p_k\}$  according to  $Score(p_i)$   
 6 **return**  $P$

**Fig. 4.** The Rank algorithm.

(boolean), Relax (boolean), that the product features in  $X$  are DestName (symbolic), RockClimbing (boolean), Hiking (boolean), and that the collaborative features of the current case  $c$  are:  $clf = (single, medium, true, true)$ . Assume that a user query  $q$  has retrieved the products:  $p_1 = (Predazzo, true, true)$  and  $p_2 = (Cavalese, true, true)$ . Then FindSimilarCases retrieves two cases  $rc_1, rc_2$ , whose similarities with the current case  $cc$  are  $Sim(cc, rc_1) = 0.75$  and  $Sim(cc, rc_2) = 1$ . Let further assume that  $rc_1$  and  $rc_2$  contain the product  $rp_1 = (Campiglio, true, true)$  and  $rp_2 = (Cavalese, true, true)$  respectively. Moreover, the products similarities are (see Section 4.3 for the similarity definition):  $Sim(p_1, rp_1) = 0.66$ ,  $Sim(p_1, rp_2) = 0.66$ ,  $Sim(p_2, rp_1) = 0.66$ ,  $Sim(p_2, rp_2) = 1$ . The score of each  $p_i$  is computed as the maximum of  $Sim(cc, c_j) * Sim(p_i, rp_j)$ , thus:  $Score(p_1) = \max\{0.75 * 0.66, 1 * 0.66\} = 0.66$ , and  $Score(p_2) = \max\{0.75 * 0.66, 1 * 1\} = 1$ . Then finally  $p_2$  is scored better than  $p_1$ .

### 4.3 Similarity Measure

For both case and product similarity we use a modified version of the Euclidean Overlap Metric (HEOM) [16]. If  $x, y \in \prod_{i=1}^n F_i$  are two generic feature vectors, then:

$$d(x, y) = \frac{1}{\sqrt{\sum_{i=1}^n w_i}} \sqrt{\sum_{i=1}^n w_i d_i(x_i, y_i)^2} \quad (2)$$

where:

$$d_i(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \text{ or } y_i \text{ are unknown} \\ overlap(x_i, y_i) & \text{if the } i\text{-th feature is symbolic} \\ \frac{|x_i - y_i|}{range_i} & \text{if the } i\text{-th feature is finite integer or real} \end{cases}$$

where  $range_i$  is the difference between the maximum and minimum value of a numeric feature  $F_i$ , and  $overlap(x_i, y_i) = 1$  if  $x_i \neq y_i$  and 0 otherwise. In addition we have integrated this metric with a cut-off concept. For each feature there is a cutoff value  $0 \leq \lambda_i \leq 1$ . If one feature distance  $d_i(x_i, y_i)$  is greater than  $\lambda_i$  then the overall distance  $d(x, y)$  becomes 1. So, for instance, if the cut-off value for the feature "cost" is 0.2 then two items having a normalized cost difference greater than 0.2 are considered as maximally distant (or minimally similar). Using cut-offs, features can be made maximally important in a non-linear way, and this cannot be achieved with a weighting schema.

If  $c = (clf, cnq, cart, v)$  and  $c' = (clf', cnq', cart', v')$  are two cases and  $p, p' \in X = \prod_{i=1}^n F_i$  are two products then:

$$Sim(c, c') = 1 - d(clf, clf') \text{ and } Sim(p, p') = 1 - d(p, p').$$

By the above definition any similarity value is in a range between 0 and 1, since the distance functions have their image values in the same range.

## 5 Evaluation Results

We have empirically evaluated the methodology introduced in section 2. We built two variants of the same showcase called ITR+ and ITR-<sup>4</sup>. ITR+ fully implements the proposed methodology and ITR- is obtained by discarding from ITR+ the interactive query management and ranking functions. In other words, the second system has almost the same GUI but is not able to recommend changes to a content-based query when a failure occurs and does not provide any sorting of the selected products (the visualization order reflects the position in the catalogue). ITR- does not use the case base of recommendation sessions, it only searches in the electronic catalogues. The two systems, as query result, present three recommendations per page. A sample recommendation page is shown in Figure 5. We randomly assigned users to ITR- and ITR+ without mentioning the fact that we were testing two variants. In fact, the tests for two user groups were performed on two different days. First, ITR- was used by 19 users and then ITR+ by 16. The ranked recommendations computed by ITR+ were based on 35 cases. Among these cases, 20 were inserted by ourselves and 15 were obtained by the ITR- trial. The users were students of the University of Trento, coming from different Italian regions and European countries, and some administrative personnel of the University. The assigned task was the same, i.e., to select a set of travel products to arrange a vacation in Trentino. Before solving the task, users took their time to learn the system without any direction from those that administered the test, they used only the instructions on the web site. A number of objective measures, which were taken by logging the user activity, are shown in Table 1. It includes the average values (in a user group) and standard deviations of the measures taken for each recommendation session. Values marked with \* (\*\*) means a significant difference at the 0.1 (0.05) probability level, according to an unpaired t-test.

<sup>4</sup> ITR+ is accessible at <http://itr.itc.it>.

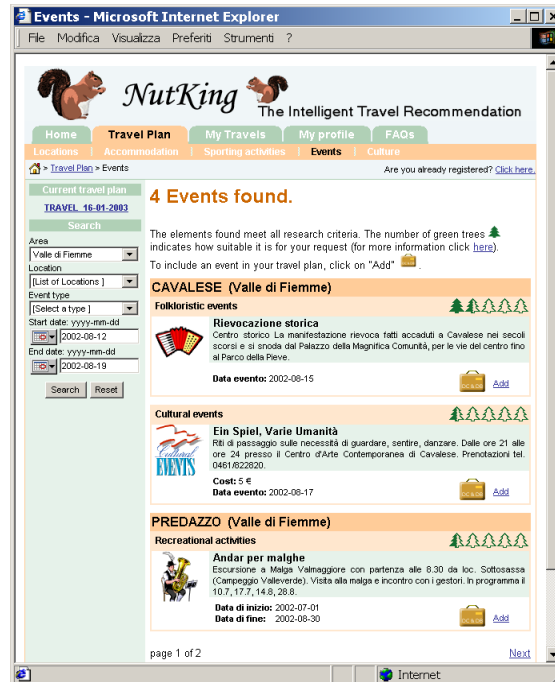


Fig. 5. Recommended events (example).

The input information provided by the two groups were essentially equal. The user provided (on average) 12.3 (ITR-) and 11.5 (ITR+) collaborative features, and in each query they constrained 4.7 (ITR-) 4.4 (ITR+) content features on average. Therefore apparently the proposed recommendation functions did not influence the users eagerness to provide information about their preferences.

The situation changes when we observe the number of queries executed and the output of the system. Users of ITR- issued a larger number of queries than those using ITR+, i.e., 20.1 vs. 13.4. Moreover, it is interesting to note that the maximum number of submitted queries for ITR- is 81 vs. 29 for ITR+. The reduction in the number of queries is due to the interactive query management support. Moreover, ITR- outputs on average a significantly larger number of results for each user query (42.0) than ITR+ 9.8. We believe that this caused the difference in the number of pages displayed by the two systems. In fact ITR- displayed 93.3 pages per session and ITR+ 71.3. In both cases the number of pages displayed is linearly correlated with the number of issued queries, with correlation coefficients: 0.67 (ITR-) and 0.78 (ITR+). It is interesting to note that the session duration is actually a bit longer for ITR+ and this fact shows that ITR+ users have devoted much more time than ITR- users to browse the information content rather than navigating through the system.

**Table 1.** ITR+ vs. ITR-, objective measures per user session.

	ITR-	ITR+
Queries issued by the user	20.1±19.2	13.4±9.3*
Content features in a query (average)	4.7±1.2	4.4±1.1
Collaborative features provided	12.3±1.4	11.5±2.0
Results per query (average)	42.0±61.2	9.8±14.3**
Page displayed	93.3±44.3	71.3±35.4**
Items in the case	5.8±3.9	4.1±3.4
Session duration (minutes)	28.5±9.5	31.0±12.3
System suggested query relaxations	n.a.	6.3±3.6
User accepted query relaxations	n.a.	2.8±2.1
System suggested query tightenings	n.a.	2.1±2.5
User accepted query tightenings	n.a.	0.6±0.9
Selected item position in the result list	3.2±4.8	2.2±2.9**

Regarding the usage of the query tightening (system suggests new features to add) and relaxation (system suggests constraints to discard or change), we note that users seem more inclined to use the relaxation rather than the tightening. The first was suggested by the system 6.3 times per session, and this means that approximately 50% of the queries had a "no result" failure. The user accepted one of the proposed relaxation 2.8 times, i.e. almost 45% of the time. We consider this a good result, taking into account the user behavior that is often erratic and not always focussed in solving the task. The corresponding results for query tightening are less convincing. A tightening suggestion was proposed by the system 2.1 times per session, which accounts for 16% of the user queries and was accepted by the user 0.6 times, i.e., 28% of the times it was suggested. We must observe that tightening was suggested by ITR+ when more than 10 items satisfied the user query. This could be a too small value, and we do not have comparisons with other tightening methods (e.g. those based on pure entropy or information gain maximization). But this result suggests that query tightening (also called attribute selection methods [13]) could have been overemphasized in the CBR literature. The impression we got from the users and from the result of a questionnaire, is that users seem to be able to refine a query when there are too many results, but it is more difficult for them to understand why a query fails to return any item.

Finally, we comment on the last measure in Table 1, namely the position of the item selected by the user in the recommended list. We recall that ITR- shows the query results in the same order as they are found in the catalogues, whereas ITR+ sorts the product using the proposed twofold similarity computation. The difference for the two systems is significant at the 0.05 level (t-test, unpaired, one-tail) and shows that ITR+ ranks the items selected by the user higher in the result list than ITR-. The average position of a selected item is 2.2 in ITR+ compared with 3.2 in ITR-. The last two numbers are computed on all the user queries that ended up with selection of an item (i.e. saved in the cart).

## 6 Conclusions and Future Work

In this paper we have presented a novel approach to products recommendation that exploits both content-based and collaborative-based filtering.<sup>5</sup> The proposed methodology helps the user to specify a query that filters out unwanted products in electronic catalogues (content-based). A human/computer interactive dialogue is managed till a reasonable amount of products is selected. Then, the selected products are ranked exploiting a case base of recommendation sessions (collaborative-based). Among the user selected items the system ranks higher items similar to those selected by other users in similar sessions (twofold similarity).

The approach has been applied to a web travel application and it has been empirically evaluated with real users. We have shown that the proposed approach: a) reduces the number of user queries, b) reduces the number of browsed products and c) the selected items are found first on the ranked list.

The future work will be devoted to the exploitation of the whole case in the collaborative stage, i.e., in using the complete case structure to match similar recommendation sessions [1]. Moreover we are investigating other recommendation methodologies that are driven by the presentation of selected examples more than by question answering [14]. The goal is to provide a range of recommendation functions for an large set of users free to adopt multiple decision styles [7].

## References

1. 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.
2. 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.
3. D. Bridge and A. Ferguson. Diverse product recommendations using an expressive language for case retrieval. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 43–57, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
4. R. Burke. Knowledge-based recommender systems. In J. E. Daily, A. Kent, and H. Lancour, editors, *Encyclopedia of Library and Information Science*, volume 69. Marcel Dekker, 2000.
5. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

---

<sup>5</sup> 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).

6. P. Cunningham, R. Bergmann, S. Schmitt, R. Traphner, S. Breen, and B. Smyth. Websell: Intelligent sales assistants for the world wide web. In R. Weber and C. Wangenheim, editors, *Procs. of the Workshop Programme at the Fourth International Conference on Case-Based Reasoning*, 2001.
7. D. R. Fesenmaier, F. Ricci, E. Schaumlechner, K. Wöber, and C. Zanella. DI-ETORECS: Travel advisory for multiple decision styles. In A. J. Frew, M. Hitz, and P. O'Connors, editors, *Information and Communication Technologies in Tourism 2003*, pages 232–241. Springer, 2003.
8. 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.
9. K. M. Gupta, D. W. Aha, and N. Sandhu. Exploiting taxonomic and causal relations in conversational case retrieval. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 133–147, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
10. F. Ricci, B. Arslan, N. Mirzadeh, and A. Venturini. ITR: a case-based travel advisory system. In S. Craw and A. Preece, editors, *6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 613–627, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
11. F. Ricci, N. Mirzadeh, and A. Venturini. Intelligent query management in a mediator architecture. In *2002 First International IEEE Symposium "Intelligent Systems"*, pages 221–226, Varna, Bulgaria, September 10-12 2002.
12. J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115–153, 2001.
13. S. Schmitt, P. Dopichaj, and P. Domínguez-Marín. Entropy-based vs. similarity-influenced: attribute selection methods for dialogs tested on different electronic commerce domains. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 380–394, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
14. H. Shimazu. ExpertClerk: Navigating shoppers buying process with the combination of asking and proposing. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001*, pages 1443–1448, Seattle, Washington, USA, August 4-10 2001. Morgan Kaufmann.
15. D. O. Sullivan, D. Wilson, and B. Smyth. Improving case-based recommendation, a collaborative filtering approach. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002*, pages 278–291, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.
16. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 11:1–34, 1997.