ITR: a Case-Based Travel Advisory System

Francesco Ricci, Bora Arslan, Nader Mirzadeh and Adriano Venturini

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

Abstract. This paper presents a web based recommender system aimed at supporting a user in information filtering and product bundling. The system enables the selection of travel locations, activities and attractions, and supports the bundling of a personalized travel plan. A travel plan is composed in a mixed initiative way: the user poses queries and the recommender exploits an innovative technology that helps the user, when needed, to reformulate the query. Travel plans are stored in a memory of cases, which is exploited for ranking travel items extracted from catalogues. A new 'collaborative' approach is introduced, where user past behavior similarity is replaced with session (travel plan) similarity.

1 Introduction

1.1 Problem Statement

There is a continuously growing number of web sites that support a traveller in the selection of a travel destination or a travel service (e.g., flight or hotel). Typically, the user is required to input product constraints or preferences, that are matched by the system in an electronic catalogue. All the major eCommerce web sites dedicated to tourism, such as Expedia, Priceline, TISCover, etc, implement this simple and quite effective pattern.

Actually, planning a travel towards a tourism destination is a complex problem solving activity. The term "destination" itself refers to a "fuzzy" concept that lacks a commonly agreed definition. For instance, even the destination spatial extension is known to be a function of the traveller distance from the destination. Italy could be a destination for a Japanese, but not for a European traveller who may focus on a specific region, such as Tuscany.

Secondly, the "plan" itself may vary greatly, in the structure and content. For instance, some people search for prepackaged solutions (all included) while other "free riders" want to select each single travel detail independently. There is a vast literature investigating: how the travel planning decision process unfolds, the main decision variables and their relationships [11]. Several choice models have been proposed [2, 14, 6] (to quote only a few). These models identify two groups of factors that impact on the destination choice: personal and travel

features. In the first group there are both socioeconomic factors (age, education, income, etc.) and psychological/cognitive (experience, personality, involvement, etc.). In the second group we could list travel purpose, travel party size, the length of travel, the distance, and the transportation mode.

1.2 Recommender Systems

The major eCommerce web sites dedicated to travel and tourism have recently started to better cope with leisure travel planning by incorporating recommender systems, i.e., applications that provide advice to users about products that might interest them in [3]. Recommender systems for travel planning try to mimic the interactivity observed in traditional counselling sessions with travel agents. The two most successful recommender systems, triplehop.com and vacationcoach.com, can be classified primary as *content-based*. The user expresses needs, benefits and constraints using the offered language (attributes) and the system matches this description with items contained in a catalogue of destinations (described with the same language). Vacationcoach exploits user profiling by explicitly asking the user to classify himself in one profile ("culture creature", "beach bum", "trail trekker", etc.) that apparently induces some implicit needs not provided by the user. The matching engine of TripleHop guesses importance for attributes not explicitly mentioned by the user, combining statistics on past user queries and a prediction computed as a weighted average of importance assigned by similar users [5]. Neither of these systems can support the user in building a "user-defined" travel, made of one or more locations to visit, an accommodation and additional attractions (museum, theater, etc.). Moreover neither of these exploit the knowledge contained in previous counselling sessions stored as cases. In fact, the application of CBR to Travel and Tourism is still in a very early stage. CABATA, which was designed as a similarity-based retrieval system, was the first attempt to apply CBR to this application domain [13].

1.3 The Proposed Approach

The ITR (Intelligent Travel Recommender) system here described incorporates a human choice model extracted from the literature specialized in the analysis of the traveller's behaviour [11, 10], and extends the quoted recommender systems. ITR supports the selection of travel products (e.g., a hotel or a visit to a museum or a climbing school), and building a *travel bag*, that is a coherent (from the user point of view) bundling of products. We call it *bag* to emphasize that we are not dealing with temporal planning or scheduling of actions in the plan. The system exploits a case base of travel bags built by a community of users as well as catalogues provided by a Destination Management Organization (APT Trentino). The proposed case structure is hierarchical [18, 19] and implemented as an XML view over a relational data base [16]. A case is decomposed into: travel wishes , travel bag, user, and reward (case outcome). Furthermore a travel bag is modelled as a tree of typed *items* (locations, accommodations, attractions, activities).

ITR integrates case-based reasoning with interactive query management, also called cooperative database research [8]. The goal is to create a system that attempts to understand the gist of a user query, to suggest or answer related questions, to infer an answer from data that is accessible, or to give an approximate response. In the CBR community this vision is shared by "conversational" approaches [1, 9]. ITR tries first to cope with user needs satisfying the logical conditions expressed in the user's query and, if this is not possible, it suggests query changes (relaxation and tightening) that will produce acceptable results. In ITR failures to satisfy all user needs are not solved relying on similarity based retrieval, as is usual in CBR. Instead, (case) similarity is exploited first, to retrieve relevant old recommendation sessions, and second to rank the items in the result set of the user's given logical query. The rank is given by computing the similarity of the items in the result set with those contained in past similar sessions. Thus the "collaborative" principle of transferring recommendations between members of a community is used, but the "correlation" between users is performed at the level of the session, hence overcoming classical limitations of Automated Collaborative Filtering (ACF) that requires user identification and a considerable amount of sessions data for each single user, before delivering effective recommendations [17]. Moreover, ITR integrates information (up to date) contained in web catalogues and "good" examples of bundling of travel products contained in previous travels built by a community of users. In this respect ITR follows an approach similar to that exploited in SPIRE [4], a system that integrates CBR with an information retrieval component.

The rest of the paper is organized as follows. Section 2 describes a typical interaction with the system and lists the major abstract functions implemented. Sections 3, 4, and 5 present the hierarchical case model, the logical components of the interactive query management component and the similarity measures exploited in the ranking method. Section 6 describes the software architecture and finally Section 7 summarize the results and points out shortcomings and limitations of the current approach.

2 Interaction with the system

This section describes a typical user/system interaction and shows some of the system functions. Let's assume that a traveller wants to have a summer vacation with his family in Trentino. He is looking for a specific location where to stay for two weeks, and for a hotel. The main page, as depicted in Figure 1, allows the traveller to input his general travel wishes and constraints. The user makes explicit the most important features that characterize the desired travel. A travel bundles tourist products (see Section 3 for the formal definition), each of them characterized by a rich set of features. Furthermore, there are additional features, more abstract and involving the whole travel that should be considered (e.g., how much experienced is the traveller in the place he is going to visit is an important variable). Among these the most important features are asked in the main page shown in Figure 1. In this scenario, the traveller specifies that he will travel with

Fig. 1. ITR's main page.

his family, that his budget is between 20 and 40 euros and he will use his car; he wants to stay in a hotel for 1 week in July, and he comes from Italy. He and his wife have never been in Trentino, and they wish to relax and to practice some sport activities.

Then the system allows the user to search and add locations and activities to the tourist's travel bag, a cart where the user stores the items that he considers interesting and he wants to consume or visit during his vacation. He can bundle his travel by seeking one or more accommodations, locations, attractions or activities. Our traveller starts looking for a specific location. The system presents a page to the user where, on the left, he can specify some logical constraints on the location. He is looking for a place where he can practice paragliding. The system searches in the locations catalogue, and suggests the three locations that match the best (Figure 2). In this scenario, Cavalese is the best choice. The system shows the rank assigned to each item, and provides an explanation (by clicking on the question mark icon). The explanation arguments is that the suggested item is similar to another item contained in other travels, built by the traveller himself or by other users, having similar wishes and constraints. Assuming that our traveller is convinced that Cavalese is the most suitable location for him, he then adds it to his personal travel bag.

Then the traveller looks for an accommodation in Cavalese. He would like a three star hotel, whose cost is between 20 and 40 euros and that accepts pets. Figure 3 shows a situation in which no accommodation that meets all the constraints is found. Hence, the system helps the user by suggesting how he can modify the query to get some results. If the user accepts to pay more

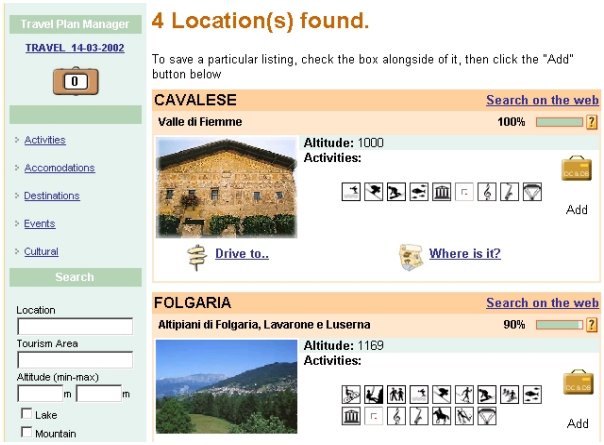

Fig. 2. Recommended locations (example).

then 40 euros he gets 8 results, and if he accepts a hotel whose category is different from 3 stars, he gets 5 hotels, etc. The traveller accepts to relax the category constraint. The system proposes a list of 5 hotels, ranked according to his wishes and constraints, which meet all the constraints but the category. The user browses the list of the suggested hotels, and adds one of them to his personal travel bag. With a similar interaction, the user can further add attractions or sports activities.

3 Case model

The Case Base (space) is made of four components: travel wishes and constraints TWC, travel bag TB, user U and reward R.

$$CB = TWC \times TB \times U \times R$$

In the following we shall describe each single component in detail. A case is built during a human/machine interaction, therefore is always a structured snapshot of the interaction at a specific time. So for instance we shall talk about partial cases to refer to those that are still under development, and simply to cases as those "completed" and stored in the case base.[1]

Figure 4 depicts an example case. Complex case components are decomposed into more elementary components in a tree-structured representation. Terminal nodes of the tree are modelled as feature vectors (items). We detail these four components in the following paragraphs.

TWC is the data structure that defines the general wishes and constraints for the given case. These are expressed as a partially filled vector of features.

[1] Actually, there is no structural definition of complete case, and this can only be considered as a dynamic role played by a case in a real human/machine interaction.

Fig. 3. Query relaxation suggestions (example).

The main features are: Travel Party (alone, with partners, a few friends, etc.); Budget (a constraint on the day allowance); Transportation mean (car, camper, bike, etc.); Accommodation Type (hotel, apartment, etc.); Departure (country of residence); Time (the period and the duration of the travel); Experience (the level of knowledge of the selected location); Goals (a list of Boolean features that must capture the most important interests of the user, e.g., sport, adventure, relax, art, culture, etc.).

In Figure 4 the example illustrates a set of travel wishes and constraints where: travel party is "family", budget is "20-40", accommodation is "hotel", and period is "July".

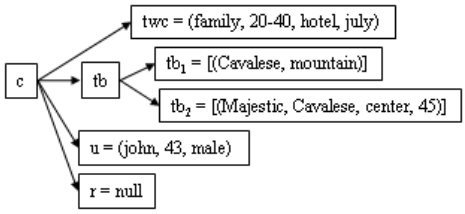

Fig. 4. An example of an ITR case.

TB is a complex data structure that collects together the items (subcomponents) selected during a case session.

$$TB = Seq(X^1) \times \cdots \times Seq(X^4) \tag{1}$$

The spaces $X^1, \ldots, X^4$ are vector spaces. $X^1, \ldots, X^4$ are item spaces: locations, accommodations, activities and attractions respectively. $Seq(X^i)$ denotes a se-

quence of elements in X^i, and this is used to model the fact that a travel bag may contain more that one item of a certain type. Travel bags are therefore elements of this type:

$$tb = ([x^{1,1}, \ldots x^{1,j_1}], \ldots, [x^{4,1}, \ldots x^{4,j_4}])$$

where each $x^{i,j}$ is a feature vector belonging to the space X^i. In Figure 4 we have:

$$tb = ([(Cavalese, mountain)], [(Majestic, Cavalese, center, 45)], \emptyset, \emptyset)$$

This travel bag contains only two items, one of type "location" and another of type accommodation. The first is $tb_1 = [(Cavalese, mountain)]$, where, for sake of simplicity, only two features are shown: location name is "Cavalese" and location type is "mountain". The second is $tb_2 = [(Majestic, Cavalese, center, 45)]$, where four features are shown: name is "Majestic", location-name is "Cavalese", near-by is "center" and cost is "45".

The user is modelled as a vector of features in similar way as an item space $U = \prod_{i=1}^{n} U_i$. Figure 4 shows an example where the user is identified with three features: name, age, and sex.[2]

The rank of a case is the evaluation of the case done by the user that created the case. The rank is structured according to the travel bag, i.e., there is a rank for the whole bag and ranks for the items in the bag. A rank is a finite subset of the integers.

$$R = RB \times Seq(RX^1) \times \cdots \times Seq(RX^4)$$

where $RB = RX^1 = \cdots = RX^4 = \{1, 2, 3, 4, 5\}$. For instance, commenting the example in Figure 4, if the user "John" had evaluated the case c we could have $r = (4, [5], \emptyset, \emptyset, \emptyset)$. This means that the whole case is evaluated 4, and the location Cavalese is evaluated 5. In this example, the user has not assigned any evaluation to the hotel. When an item is not explicitly evaluated, the average value 3 is assumed as default value. Hence, if a component was not rated by the user it gets the rate 3 because we assume that the inclusion in the bag represents a rather positive evaluation. Values below 3 are considered negative, so the user can provide both positive and negative feedback.

We conclude this section by commenting again the structure of a case and the containers of the data. Figure 5 depicts the relationships between the case base and the catalogues of items. The case base contains the structured cases, which are represented as hierarchical data structures. Some case terminal nodes, such as those that are items included in the travel bag, are in fact pointers to objects contained in catalogues of locations, accommodations, activities and attractions. Both case base and catalogues are exploited in our CBR approach. The case base provides information about good bundling of products and is therefore used for learning this knowledge and for ranking items selected in the catalogues. The catalogues are exploited for obtaining up-to-date information about currently available products.

[2] The full list contains additional features, like country or profession, but in fact this data is never used by ITR for personalization purposes.

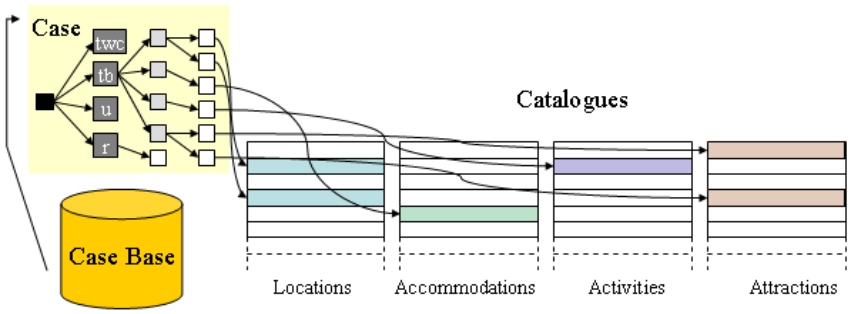

Fig. 5. Case base and catalogues.

4 Interactive Query Management

In this Section we describe the basic data structures and the functions designed to interactively search in a catalogue of items. Figure 6 shows the main logical architecture designed to help the user to satisfy his information goal. The client application (typically the Graphical User Interface) interacts with the Intelligent Mediator (IM) that, if it is possible, will retrieve a reasonable set of items that meet the query, or else suggest to the client some refinements to the query. The Intelligent Mediator in turn interacts with a data processing engine to fetch data (result set)[16]. More details on the interactive query management functions supported by the IM and the exact specification of the algorithms, which are here only sketched, can be found in [15].

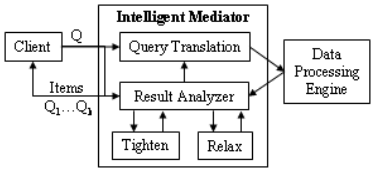

Fig. 6. Intelligent Mediator Architecture.

The information needs are encoded as a query, say Q, and the reply of IM to Q could take the form of a set of items extracted from the product catalogues or a new set of queries. The reply is a set of items when the Intelligent Mediator decides there is a satisfactory result for the query in the product catalogues, otherwise, when a "failure" situation occurs, it suggests a new set of alternative queries that, by tightening or relaxing some of the query constraints, might produce satisfactory results.

As we said above, a travel bag is made up of travel items belonging to four different types (location, accommodation, activity and attractions). We shall

denote by X an item space (of a particular type), and this is a vector space $X = \prod_{i=1}^{n} X_i$, i.e., an item of type X is represented as a $n-$dimensional vector of features. We consider three general types of features: finite integer, real and symbolic. Given an item space $X = \prod_{i=1}^{n} X_i$, we shall say that the set $CX \subset X$ is a *catalogue* of type X.

The query language, which is used to search in the catalogues of items, is quite simple, but this makes possible the interactive query management described later in this section. A query Q over a catalogue $CX \subset X$ is obtained by the conjunction of simple constraints, where each constraint involves only one feature. More formally, $Q = C_1 \wedge \cdots \wedge C_m$, where $m \leq n$, constraint C_k involves feature x_{i_k}, and:

$$C_k = \begin{cases} x_{i_k} = v_k & \text{if } x_{i_k} \text{ is symbolic} \\ l_k \leq x_{i_k} \leq u_k & \text{if } x_{i_k} \text{ is finite integer or real} \end{cases} \quad (2)$$

where v_k is a possible value for the $k-$th feature, i.e., $v_k \in X_k$ and $l_k \leq u_k$ are the boundaries for the range constraint on the feature, and $l_k, u_k \in X_k$.

Query relaxation changes a query definition in such a way that the number of items returned by the query is increased. A query is typically relaxed when the result set retrieved from the item space X is void, or when the user is interested in having more examples to evaluate. The relaxation process implemented by the relax component of the Intelligent Mediator is shown in Figure 7.

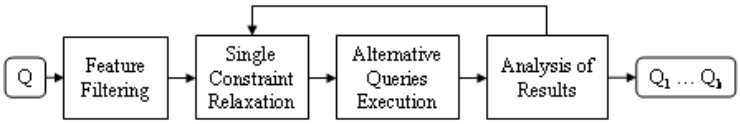

Fig. 7. ITR's query relaxation process.

The query relaxation component takes in input a query $Q = C_1 \wedge \cdots \wedge C_m$ and builds a new set of relaxed queries $Q_1, \ldots Q_k$, such that each Q_i relaxes only one constraint of the original query. In the following discussion we shall use the query $Q = ((x_1 = ski) \wedge (x_2 = Cavalese) \wedge (x_3 = \top) \wedge (x_4 \leq 100))$, where we are searching in a database of sport activities, and the features are $SportType, Location, School$, and $Cost$ (and $\top$ means "true"). The relaxation is computed in the following way:

1. **Features Filtering.** This module identifies those features' constraints that cannot be relaxed without changing the information goal of the user. Referring to the example above, the user might well be interested in some school a bit more expensive or maybe not in Cavalese but he is interested in skiing, not in fishing.
2. **Single Constraint Relaxation.** Whenever the relaxable constraints have been identified, for each of them a "relaxed" version (C_i') must be identified. Two different approaches are exploited according to the feature type:

symbolic features constraint are relaxed by simply discarding it; in numeric feature constraints the range of allowed values is increased by a percentage that is feature dependent. After this stage the relaxed constraints in Q are: $x_2 = ALL$, $x_3 = ALL$ and $x_4 \leq 110$.

3. **Alternative Queries Execution.** At this stage, for each "relaxable" constraint a new relaxed version of the original user's query is built, with only that constraint relaxed. In our example we have three new queries: $Q_1 = ((x_1 = ski) \wedge (x_2 = ALL) \wedge (x_3 = \top) \wedge (x_4 \leq 100))$, $Q_2 = ((x_1 = ski) \wedge (x_2 = Cavalese) \wedge (x_3 = ALL) \wedge (x_4 \leq 100))$, and $Q_3 = ((x_1 = ski) \wedge (x_2 = Cavalese) \wedge (x_3 = \top) \wedge (x_4 \leq 110))$. These relaxed queries are executed and the number of items retrieved by each single query is determined (counts).
4. **Analysis of Results.** When the results (counts) are obtained they must be analyzed. The goal is to understand if the relaxed queries have produced an improvement, i.e., if the new set of results (for each query) is not still void or become too large. If this last situation occurs a tightening is required.

Having described the query relaxation issues, we now illustrate the query tightening process accomplished by the tighten component of the Intelligent Mediator (Figure 6). When a query returns too many items, i.e., above a certain threshold α (in ITR this has been set to 20), the system finds some ways to improve the user's query, i.e., it suggests additional features to constrain.

The query tightening process exploits two feature ranking methods to select the features to be suggested to the user for tightening the result set. The choice of one method relies on a second threshold parameter β: *if the result set cardinality of the input query Q is above β then the* **Off-Line-Entropy** *method is used, otherwise the* **On-Line-Entropy-Mutual-Info** *is applied*. We describe in more detail the two methods below:

Off-Line-Entropy : this method uses the entropy of a feature, computed using the distribution of feature values found in the catalogue:

$$H_S(X_i) = -\sum_{v \in X_i} p(v) \log[p(v)], \tag{3}$$

where X_i is a feature space and $p(v)$ is an estimate of the probability to find the value v for the i-th feature in the sample S (the full catalogue). The top three features, not already constrained in Q, are then suggested to the user as additional features to be constrained. Feature entropy provides a rough estimate of feature selectivity, and requires no computational cost as these values are precomputed. This has proved to be satisfactory when there is no need to be precise since the current result set is very large and selective features are easy to find.

On-Line-Entropy-Mutual-Info : this method is much more expensive, with respect to the computational cost, and therefore is applied only when the size of the results set is smaller (less than the quoted β threshold). If we denote with

S_Q the result set of Q, then for each feature X_i, not already selected in Q, the score is computed as:

$$\frac{H_{S_Q}(X_i)}{1 + max\{H_{S_Q}(X_i; X_j) | X_j \in SF\}} \tag{4}$$

where SF is the set of already selected features (those constrained in Q), and the mutual information between X_i and X_j is defined as:

$$H_{S_Q}(X_i; X_j) = H_{S_Q}(X_i) + \sum_{v \in X_i,\, u \in X_j} p(v, u) \log[p(v|u)] \tag{5}$$

This score method tends to prefer features that have a large entropy and are not correlated with those already constrained in Q. In a similar manner as above, the top three features are proposed to the user for constraining the original query. (the reader is referred to [15] for a detailed description of the feature selection algorithm).

5 Item Ranking

We now describe how similarity based scoring is performed when items retrieved from the catalogues are recommended. Basically, in this situation we have two subsets of items $X \supset S, R$. $S = \{x_1, \ldots, x_m\}$ is the set of items that must be ranked and R is the (reference) set of items that are used to assign scores to elements in S. S is the result of a standard query that is eventually defined incrementally by the user using the support of the intelligent mediator (as described above). The reference set is made of those items, of the same type as those in S, extracted from a references set of case, i.e., cases that are similar to the current recommendation session. For this reason, our ranking approach can be defined as item-based, but in contrast with other approaches in this category [17], we do not compare the items to be recommended with those previously selected by the same user for which the recommendation is to be built. We use instead a dynamic set of items contained in the travel bags of different users whose specific context data (travel wishes and constraints as defined in Section 3) are similar to the current situation.

The ranking process is illustrated in Figure 8. In this example Q is a query on the location items catalogue that contains the user-specific wishes. Let us assume that the locations in $S = \{loc_1, loc_2, loc_3\}$ are retrieved. Then using the current definition of the case the top ten cases are retrieved from the case base. In this stage the current definition of travel wishes/constraints are used to retrieve the 10 most similar cases (reference cases set).

Finally the last step is ranking these locations according to the similarity to the locations R contained in this set of retrieved cases. This is performed with a call to similarity-based sorting function that ranks best the locations in S that are closest to the location in R (distance between a point and a set). The metric

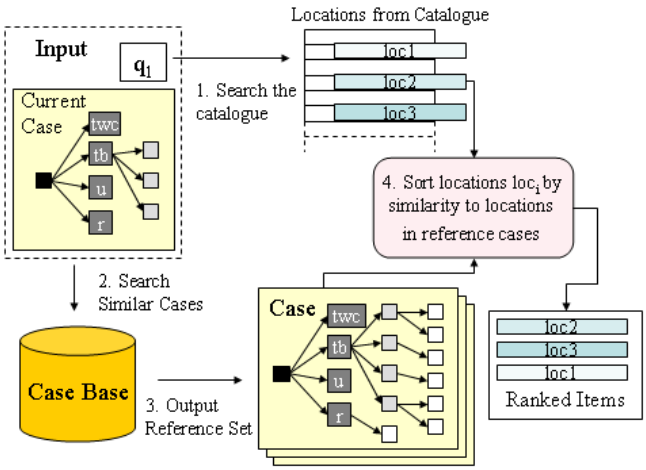

Fig. 8. Sorting items extracted from a catalogue.

used both for case and item similarity is the Euclidean Overlap Metric (HEOM) [21]:

$$heom(x,y) = \frac{1}{\sqrt{\sum_{i=1}^{n} w_i}} \sqrt{\sum_{i=1}^{n} w_i d_i(x_i,y_i)^2} \tag{6}$$

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, and $overlap(x_i, y_i) = 1$ if $x_i \neq y_i$ and 0 otherwise. The weights $0 \leq w_i \leq 1$, $i = 1, \ldots, n$ are assigned dynamically according to the query definition Q: the features used in Q receive a larger weight (the other features are weighted proportionally to the frequency of usage in the reference set of cases). When this metric is used for case similarity only the travel wishes are considered (see Section 3).

At this stage the user can add one of those recommended items to the travel bag, or change some query constraints and re-run the ranking process. Items are added by the user and is up to him to maintain any sort of logical consistency within the bundling. Therefore, one user may add two locations that for a second user are too far or a selection of activities that seems unreasonable to another. Obviously, this is a limitation of the proposed approach, but since no simple solution is applicable, we decided to acquire a set of real cases before designing a viable solution to this problem.

6 Software Architecture and Implementation

This section describes the software structure of the implementation that is based on the Java 2 Enterprise Edition Architecture [12], and it uses a thin client approach (html only). Figure 9 shows the main software components.

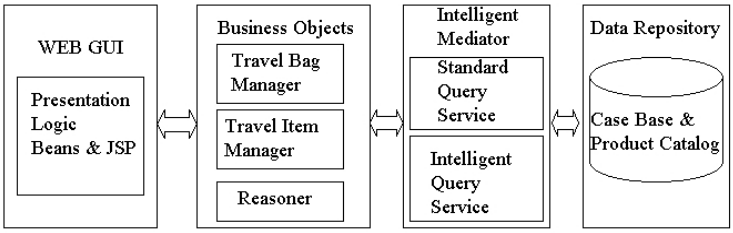

Fig. 9. ITR software architecture.

Web GUI This manages the user interaction by dynamically constructing the user interface and interpreting the user commands. This layer is built upon the Struts framework, an open source jakarta project that helps the adoption of the MVC pattern (http://jakarta.apache.org/struts/). The View role is played by a set of JSP pages, while the Model and the Controller roles are performed by a set of Java beans that manage the data to be displayed, and execute the commands issued by the user.

Business Objects It includes the Travel Bag Manager, whose purpose is to store, retrieve and update travel bags; the Travel Item Manager, which allows to handle the tourist items to be shown; the Reasoner, which manages the intelligent query management and ranking process.

Intelligent Mediator It includes the Standard Query Service and the Intelligent Query Service. The first one provides basic query functions for retrieving and storing data from/to the repository as XML documents. The second one extends the Standard Query Service to support different ways of querying and ranking documents. It implements the ranking method described in Section 5, and the query management described in Section 4.

The Mediator component exploits a simple mediator architecture [20, 7] (more detail in [16]). Its purpose is to access the information stored in the repository (a relational database system) and to produce XML documents. The data integration needs have been solved in two steps. First, we have defined a set of SQL views over the complete data model (approximately 100 oracle tables). Second, for each view the corresponding XML schema has been defined. The XML documents are produced according to these schemas. A custom query language has been developed to query the XML views.

7 Conclusions and Future Work

In this paper we have proposed an approach for improving standard eCommerce applications for travel products selection and booking by leveraging more realistic and therefore complex human choice models. This is the topic of two research projects[3] being conducted at eCTRLs.

Among the major advantages of our approach we could mention:

- *The system can bootstrap easily as it is based on both a case base and a collection of available catalogues.* The recommendation is not built uniquely on reuse of past cases. A travel bag can be built "manually" by inserting services/products (items) searched in catalogues.
- *User need not be registered.* The similarity of the user's session specific goals and constraints with those contained in previous cases is used to determine the set of past experiences that must influence the recommendation.
- *Man/machine interaction supports both alternative expansion and reduction of the results.* Navigation in the information space is supported by a tool that suggests additional constraints or those that should be discarded in order to maintain the number of recommendations around a reasonable size.
- *The system does not assume that products and users shares common features.* This is the idea at the base of content-based filtering: the products are selected when their features match with those in the user's description. Recommenders are typically built in this way. Conversely, our user profile is very limited, and therefore can be easily acquired.

The ITR system is still in an early validation stage. The prototype has been built but no serious validation has been conducted. A user group (experts in the domain) has been involved in the requirements analysis stage and their input has influenced the cyclical revision of the user interface. We plan to conduct an extensive evaluation with a larger user group and to acquire a set of cases that will extend the initial case base built artificially using the travel offers of the Trentino local tourist board. Further work will address the weighting schemas and the tuning of the case similarity metric.

References

[1] D. Aha and L. Breslow. Refining conversational case libraries. In *Case-Based Reasoning Research and Development, Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR-97)*, pages 267–278. Springer, 1997.

[2] P. K. Ankomah, J. L. Crompton, and D. Baker. Influence of cognitive distance in vacation choice. *Annals of Tourism Research*, 23(1):138–150, 1996.

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

[3] 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).

[4] J. J. Daniels and E. L. Rissland. What you saw is what you want: Using cases to seed information retrieval. In *ICCBR*, pages 325–336, 1997.

[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] D. R. Fesenmaier and J. Jeng. Assessing structure in the pleasure trip planning process. *Tourism Analysis*, 5:13–17, 2000.

[7] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide web:a survey. *SIGMOD Record*, 27(3):59–74, 1998.

[8] T. Gaasterland, P. Godfrey, and J. Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1(2):123–157, 1992.

[9] 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.

[10] K. Grabler and A. Zins. Vacation trip decision styles as basis for an automated recommendation system: Lessons from observational studies. In *Proceedings of the ENTER 2002 Conference*, Innsbruck, Austria, January 22-25 2002. Springer Verlag.

[11] Y.-H. Hwang, U. Gretzel, and D. R. Fesenmaier. Behavioral foundations for human-centric travel decision-aid systems. In *Proceedings of the ENTER 2002 Conference*, Innsbruck, Austria, January 22-25 2002. Springer Verlag.

[12] N. Kassem and the Enterprise Team. *Designing Enterprise Applications with the JavaTM 2 Platform, Enterprise Edition*. Addison-Wesley, 2000.

[13] M. Lenz. Imtas - intelligent multimedia travel agent system. In *Information and Communication Technologies in Tourism (Proceedings of ENTER-96)*, pages 11–17. Springer, 1996.

[14] L. Moutinho. Consumer behavior in tourism. *European Journal of Marketing*, 21:2–44, 1987.

[15] F. Ricci, N. Mirzadeh, and A. Venturini. Intelligent query managment in a mediator architecture. In *IEEE International Symposium "Intelligent Systems'*, Sunny Day, Bulgaria, September 10-12 2002.

[16] F. Ricci, N. Mirzadeh, A. Venturini, and H. Werthner. Case-based reasoning and legacy data reuse for web-based recommendation architectures. In *Proceedings of the Third International Conference on Information Integration and Web-based Applications & Services*, pages 229–241, Linz, Austria, September 10-12 2001.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW10 Conference*, pages 285–295, Hong Kong, May 1-5 2001. ACM.

[18] B. Smyth and M. T. Keane. Design a la deja vu: Reducing the adaptation overhead. In *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press/MIT Press, 1996.

[19] A. Stahl and R. Bergman. Applying recursive cbr for the customization of structured products in an electronic shop. In *Advances in case-based reasoning: 5th European workshop, EWCBR-2000, Trento, Italy, September 6–9, 2000: proceedings*, pages 297–308. Springer, 2000.

[20] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[21] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 11:1–34, 1997.