

SUPPORT INTERACTIVE QUERY REFINEMENT FOR A TRAVEL RECOMMENDATION SYSTEM

**Adriano Venturini, Nader Mirzadeh,
Francesco Ricci, Hannes Werthner**

*eCommerce and Tourism Research Laboratory, ITC-irst, Trento, ITALY
e-mail: {ricci,mirzadeh,venturi,werthner}@itc.it*

Abstract

This paper describes a Web recommendation system that supports travellers in the selection and comparison of travel products provided by a regional tourism organisation. In addition to traditional information search and filtering, the system supports interactive query management to help its users to plan a leisure trip. Case Base Reasoning and similarity base retrieval techniques are used for selecting and ranking suggested travel services (e.g., accommodations or sport activities) as well as innovative advanced techniques to help the user to refine the query in the case it fails to satisfy the user's information needs.

Keywords: *Recommendation System, Case-Based Reasoning, Trip Planning, Mediator Architecture, XML*

INTRODUCTION

One of the main tasks of a regional tourism organisation (RTO) is to provide travellers with all the information needed to plan a travel. Even though the Web sites published by these organizations allow the user to access a complete set of information related to a destination, the comparison and the selection of tourist products are often difficult due to the amount of available items and the amount of features that have to be compared. The traveller who wants to plan a travel has to select some locations where he would stay, and some services such as sport or cultural events that he would consume. We shall refer to a location, a cultural or sport event, or a lodging service simply as a travel item (TI) or an item to emphasize all services or products that the user can select in a catalogue and add to his travel plan.

A RTO plays an important role in the promotion of a destination by providing all the information related to a place or a service. These organisations can provide information to the travellers. But, they cannot give any suggestions since they do not have enough information about the traveller's preferences, and sometimes because of some domestic laws (e.g., to be fair to the service providers).

This paper describes a Web application aimed at promoting alpine destinations. It has been developed in a collaboration between our research laboratory and the RTO Trentino¹. It assists its users to plan their travels by providing intelligent query mechanism for finding TIs that are most desirable during their travels. When the user submits his inquiry, if it produces no or too many results, the system would help him in a “conversational” way to refine the query. If a query returns a result set to a user, it is ranked according to the user's preferences.

The system uses Case-Based Reasoning technology to build a recommendation (see [Ricci01b] for more details). Case-Based Reasoning (CBR) is a problem solving methodology that faces a new problem or situation by first retrieving a past, already solved similar case, and then reusing it for solving the current problem [Aamodt94]. In our system, the cases are travel plans with their corresponding TIs built by the users. We reuse case-base to suggest additional TIs that can be interesting to the user, and rank the items returned by the user's inquiry. In addition to the CBR technique, the system adopts intelligent query refining techniques to help the user in the selection of his destination. In fact, it is often difficult to formulate a query that can retrieve a right amount of items. If the user adds too many constraints, it can happen that no item is selected, on the contrary, if it is not much selective the items selected are too many and their comparison is often difficult. Differently from what happens in normal commercial tourist web sites, our application helps the user by suggesting how the query can be modified to get a reasonable amount of items. This paper mainly focuses on this aspect, and describes how the interaction with the user is driven and the relaxation techniques implemented by the system.

The next section describes a sample scenario where a traveller using the system plans his travel by looking for a location at a certain altitude where he can practice rock climbing and rafting. The Software Architecture section explains how the software is structured. The Supporting Relaxation Query Refinement section describes the query processing and the relaxation steps. The state of the art in recommender system is outlined in the Related Work section followed by the conclusion.

SYSTEM USAGE SCENARIO

In this section we describe how a user will typically interact with the system to select a destination. A traveller has some constraints (e.g. time or budget) and preferences (e.g. location or some activities) in mind when he plans a travel, and the system should be able to suggest TIs that are tailored for this user taking into account those preferences and constraints. For instance, a traveller may search for a location that is characterized by a set of features, like altitude, inhabitants, sport facilities, historical or archaeological places, etc.

¹ APT Trentino: Azienda per la Promozione Turistica del Trentino

If the user's query returns a reasonable amount of locations, a ranked list is shown. But, it may happen that no place can be found satisfying all the constraints specified by the user.

The scenario below describes how the user could search for a location of specific altitude where he can practice some sport activities, i.e. rock climbing and rafting.

- A user searches for a location by specifying altitude not higher than 1000 meters where he can practise rock climbing and rafting.
- Suppose now, the search returns no result because there is no location with the desired properties (i.e. specified altitude, rock climbing, and rafting).
- The system will try to relax some constraints in the user's inquiry.
- The following information is computed by the system.

FEATURE	SUGGESTED ACTION	LOCATIONS
Rock Climbing	Discard	7
Rafting	Discard	20
Altitude	Below 1100	5

Table 1 – Relaxation Table

These data supply the user with enough information about the relaxation actions, and the possible result size. The first row shows there are 7 places if the rock climbing is removed from the original query, and the third row shows that there are 5 locations where he can practice his desired activities but the altitude has to be changed (relaxed) to 1100. Let's assume he decides the last option.

- The system executes the new query, and shows the ranked result set.
- The user goes through the found items.
- The user selects a location and adds it to his travel plan.

The above scenario illustrates how the user could find some locations where he can perform some sport activities. In a similar scenario, the user can search and find some lodgings in a location.

SOFTWARE ARCHITECTURE

In this section, first the main software components of the application are described and then a description of the infrastructural technologies used by the system is given. Figure 1 shows

its main components.

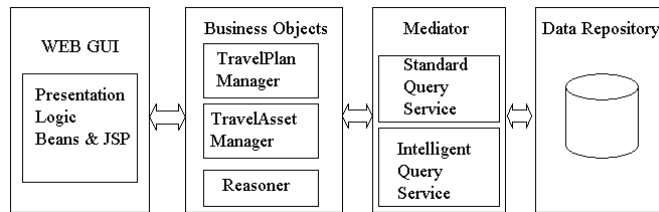


Figure 1 – Software Architecture

- **Web GUI.** This component interacts with the user by dynamically constructing the user interface and by interpreting the commands entered by the user. It is structured according to the model view controller pattern [Gamma95]. 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 (the model), and execute the commands issued by the user (the Controller) by invoking the appropriate business object methods.
- **TravelPlan Manager.** This business object executes the operations related to the management of travel plans. It interacts with the Mediator component to store, retrieve and update travel plans. Its methods are called by the GUI component to execute the functions requested by the user.
- **TravelAsset Manager.** This component provides functions for the management of the travel items. The GUI uses this component to search for travel items, such as locations or services. It interacts with the Mediator to access the data stored in the repository.
- **Reasoner.** This component provides the reasoning functions used by the TravelPlan and the TravelAsset manager components to find and suggest items to the user, or to manage query refinement process.
- **Standard Query Service.** This service (implemented in the Mediator) provides basic query functions for retrieving and storing data in the repository in form of XML documents.
- **Intelligent Query Service.** This service extends the Standard Query Service to support different ways of querying and ranking documents. It implements the similarity search, as well as the relaxation technique explained in the Supporting Relaxation Query Refinement section.

The Mediator component exploits a simple mediator architecture [Wiederhold92,

Florescu98]. It has been described in more detail in [Ricci01]. 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 RTO Trentino data model) that provides the target data model used by the system. Second, for each view a corresponding XML schema [Fallside01] has been defined, and XML documents are produced according to these schemata. A custom query language has been developed to query the XML views [Ricci01c]. Then, the Mediator receives queries as XML documents, and translates them to their corresponding SQL queries at runtime to retrieve data from physical database. These issues will be discussed in the next section.

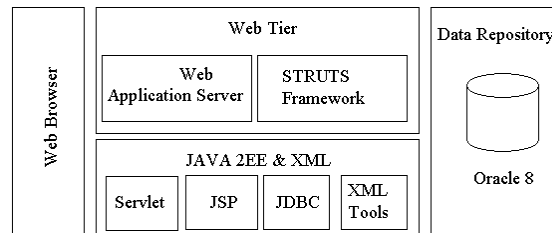


Figure 2 – Infrastructural View

Figure 2 depicts the infrastructural architecture of the recommendation system. The reference application model is the Java 2 Enterprise Edition Architecture [Kassem00]. The user interacts with the system by using a Web Browser. In the Web Tier we use Struts, an open source framework useful in building web applications with Java Servlet and JavaServer Pages (JSP) technology. Struts encourages application architectures based on the Model-View-Controller (MVC) design paradigm (<http://jakarta.apache.org/struts/>). As application server we have chosen Tomcat (<http://jakarta.apache.org/tomcat/>), the reference implementation for the JSP and Servlet technologies. The JDBC API is used by the mediator to access the information from the repository. The Xerces XML parser and the Xalan XSLT processor (<http://xml.apache.org/xalan-j>) are used by all the application objects to read, write and manipulate XML files.

SUPPORTING RELAXATION QUERY REFINEMENT

In this section we focus on query refinement, relaxation, and reasoning techniques that support the scenario given in the System Usage Scenario section. A query contains a set of constraints on features (e.g., altitude, rock climbing, and rafting features in the scenario) that characterise items to be retrieved. It may happen that a query does not return any result because there is no item that satisfies all the constraints. But the query can succeed by

applying only some slight modifications. The problem is that a user doesn't know how to modify the initial query to get a set of items that at least satisfy partially the user's needs. In our system, the reasoner component provides this type of information, and the way it finds that information is outlined below.

Each item (e.g., location) is described by a feature vector belonging to an n -dimensional space. Referring to the example from the previous section, the query contained constraints on altitude, rock climbing and rafting features of a location. The diagram below shows an example of a possible data distribution of the location.

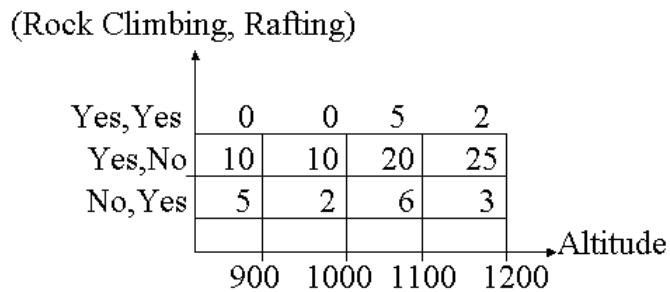


Figure 3 – Location Data Distribution

The X axis represents the altitude of locations, and the Y axis represents the combination of two features, namely rock climbing and rafting activities. Numbers next to the (x,y) points represent the number of locations that match the given combination of attributes-values. As you can see from the diagram there is no location at 1000 meters altitude with rock climbing and rafting possibilities, but there are some at 1000 meters where it is possible to practise one of the two activities, or there are some locations at higher altitude where both activities can be performed.

The relaxation procedure should try to find alternatives to the initial query. Each alternative query considers relaxation on some features while keeping the others unchanged. In general, if a query contains n constraints, the number of all possible alternatives to be evaluated is 2^n . Thus, both computation may be too expensive and the number of suggestions would likely be unacceptable by the user. In our approach, we consider relaxation on one feature at a time. Thus, if a query contains n constraints, the system would suggest at most n possible relaxations. This makes the relaxation algorithm simpler and faster, even if some better alternatives might be ignored. Note that some features cannot be relaxed depending on the type of the query. For example, if a traveller is looking for a river where he can practice rafting, the system will not consider any relaxation on rafting since it is the main objective of this query.

The reasoner also provides for each suggested relax condition the cardinality of its result

set, so based on these data the system is able to suggest how to relax the initial query to get at least some results. This is obtained by getting the result size of each relaxed version of the initial query using SQL count function. Although, in general the reasoner may issues n (one for each relaxed constraints) count queries.

From a technical point of view, relaxing a constraint on a boolean feature is here implemented discarding it, but for numeric features the reasoner finds a new wider range. At the moment we let the reasoner try a set of values to find the first shortest range that satisfies all the constraints (e.g. the altitude constraint is widened to 1100). It means the reasoner tries to extend the initial range by using the closest points from a predefined set till the new found range would satisfy all the constraints, and the process stops after a number of tentative tries is reached.

Figure 4 shows the sequence of the operations executed by the system to find some locations satisfying the user’s query. There are three actors. The user who interacts with the system via a Web interface. The Graphical User Interface (GUI) and the Reasoner which reads the submitted input to the GUI and manages the query processing. The Mediator interacts with the repository and provides the requested data to the GUI&Reasoner in XML format. The sequence of the operations is described next. Features are in **boldface** and values in *italic*.

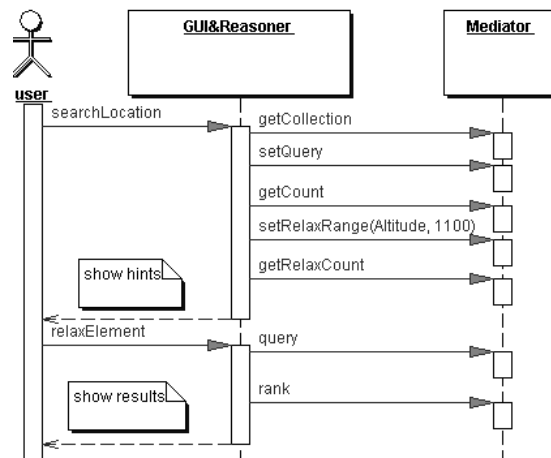


Figure 4 – Relaxation and Reasoning

1. searchLocation: The user looks for a location by specifying *1000 meters* for **altitude**, and *yes, yes* for **rock climbing** and **rafting** for activities.
2. GUI&Reasoner asks the Mediator for the location collection. Then it prepares an XML document describing the user's query, and submits it to the collection by calling the setQuery method.

3. GUI&Reasoner calls the `getCount` method to check the cardinality of the result set. We have assumed it returns an empty set in the example.
4. GUI&Reasoner discovers that altitude feature is numeric and a range constraint is given. For this constraint, it finds a new range to be tried, and submits it to the Mediator.
5. GUI&Reasoner calls the `getRelaxCount` method to return the values about relaxation (the relaxation table given in Table 1).
6. GUI&Reasoner tells the user there is no location satisfying his query, and the relaxation information is shown.
7. The user decides to relax the **altitude**. In other words, he accepts the suggested range
8. GUI&Reasoner submits the new query to the Mediator, and receives all locations. The similarity technique [Ricci01c] is used to rank the locations according to the user's preferences.

To rank items using the similarity technique, the system has to identify a set of items which represent the user's preferences and can be used as 'reference' items. At the moment we are using items from user's past travel plans, but we will investigate how to use other users' travel plans. Then, the similarity between the reference set and the found items is computed. Items are ranked according to the computed similarity. See [Ricci01c] for a full explanation of the CBR approach and similarity technique.

Related Work

Our project faces many important research issues that range from Case Base Reasoning, to indexing techniques and XML data integration. The main concern of this paper is query refinement.

There are some commercial sites that help users to find suitable trips or travel services (e.g. lodging). These web sites (e.g. Expedia.com or Travelocity.com) enable their users to define a new trip by selecting a flight, a hotel, or a car.

These web sites are not able to sort the found items according to a user's preferences even though he has entered his preferences *explicitly*, neither to help the user to suggest alternative queries in the case it fails to return any item. We provide these functions in our system.

Commercial databases support only exact match queries, meaning that just those items that satisfy completely the query are returned, and in the case of failure no item even very

“close” to the desired one will be returned. There have been many research activities to overcome this short coming by using k -nearest neighbour techniques when users would accept non exact matches close to their specification [Chaudhuri99]. Our approach is different in that we retrieve first all items fully satisfying the query. In the case no item can be found, the system is able to suggest alternative queries. CBR is used for learning user preferences, and rank recommended items.

Applying CBR to the travel domain is quite a novel approach to travel recommendation. CABATA [Lenz96, Lenz99] has been the first system that applied CBR to this domain. Our system differs significantly from CABATA in the case structure, similarity match for ranking (sorting) the found items, and the query refinement. [Ricci01,Ricci01b] contain more detail about these issues. We use XML as a language for case representation. Shimazu [Shimazu98] has been among the first to explore XML in CBR applications. Our work has been also influenced by Sengupta et al. [Sengupta 99] and Hayes et al. [Hayes99] (see [Ricci01] for additional details).

There are some commercial products (e.g. Tamino, Excelon) that transfer XML documents to and from a relational database systems. But there are still many open points, like the query language and the XML-relational mapping due to the difference of the underlying models [Baru99, Carey00, Chamberlin00, Florescu99]. Our approach is to use the power of the RDBMS to produce views that have a structure close to the underlying data model, and generating default XML documents corresponding to these views. Further, we want to investigate how intelligent query mechanisms over XML data representation can be applied.

Conclusion

This paper describes a web application aimed at supporting travellers in the process of planning a leisure travel. We would like to stress here that we are working on a real application that uses actual data provided by the Trentino RTO. This makes the development more difficult, because of data incompleteness or inconsistency. Moreover, the prototype will be tested with real users and in the real situations. In the future work, we shall study and validate more efficient and effective query refining techniques by using multi-dimensional data distribution for numeric features and bitmap indexing for boolean features. We will also use CBR and similarity techniques to suggest additional services that can meet user desires which are not explicitly requested by the user.

Acknowledgements

This project is partially funded by Fondazione Cassa di Risparmio di Trento e Rovereto.

REFERENCES

- [Aamodt94] Aamodt, A. and Plaza, E. (1994). *Case-based reasoning: foundational issues, methodological variations, and system approaches*. AI Communications, 7(1):39-59.
- [Baru99] Baru, C., Gupta, A., Lundäscher, B., Marciano, R., Papakonstantinou, Y., and Velikhov, P. (1999). *Xml-based information mediation with MIX*. In Exhibitions Program of ACM SIGMOD 99.
- [Carey00] Carey, M., Florescu, D., Ives, Z., Y. Lu, J. S., Shekita, E., and Subramanian, S. (2000). *Xperanto: Publishing object-relational data as xml*. In Proc. of the Int. Workshop on Web and Databases (WebDB).
- [Chamberlin00] Chamberlin, D., Robie, J., and Florescu, D. (2000). *Quilt: An xml query language for heterogeneous data sources*. In WebDB 2000, Dallas
- [Chaudhuri99] Chaudhuri, S., Gravano, L.. *Evaluating top-k selection queries*. In VLDB 1999, Edinburgh
- [Fallside01] Fallside, D. C. (2001). *Xml schema part 0: Primer*. <http://www.w3.org/TR/xmlschema-0/>. W3C Proposed Recommendation
- [Florescu99] Florescu, D. and Kossmann, D. (1999). *Storing and querying xml data using an rdbms*. IEEE Data Engineering Bulletin, 22(3):27-34.
- [Florescu98] Florescu, D., Levy, A., and Mendelzon, A. (1998). *Database techniques for the world-wide web:a survey*. In Proceedings of the ACM Sigmod-98. ACM press
- [Gamma95] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns*. Professional Computing Series. Addison Wesley Longman, Reading, Mass.
- [Hayes99] Hayes, C. and Cunningham, P. (1999). *Shaping a cbr view with xml*. In Althoff, K.-D., R. B., and L.K., B., editors, Case Based Reasoning Research and Development, Proceedings of the International Conference on Case Based Reasoning (ICCBR), Monastery Seon, Germany. Springer Verlag.
- [Kassem00] Kassem, N. and the Enterprise Team (2000). *Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition*. Addison-Wesley
- [Lenz96] Lenz, M. (1996). *Imtas - intelligent multimedia travel agent system*. In Proceedings of ENTER-96. Springer Verlag.
- [Lenz99] Lenz, M. (1999). *Experiences from deploying cbr applications in electronic commerce*. In Proceedings of GWCBR-99.

- [Ricci01] Ricci, F., Mirzadeh, N., and Werthner, A. V. H. (2001). *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, Linz, Austria.
- [Ricci01b] Ricci, F. and Werthner, H. (2001). *Case-based destination recommendations over an xml data repository*. In Enter2001, Montreal, Canada.
- [Ricci01c] Ricci, F. and Werthner, H. (2001). *Case-based querying for travel planning recommendation*. Information Technology and Tourism. to be published.
- [Sengupta et al,1999] Sengupta, A., Wilson, D. C., and Leake, D. B. (1999). *On constructing the right sort of cbr implementation*. In CBR workshop at IJCAI'99.
- [Shimazu98] Shimazu, H. (1998). *A textual case-based reasoning system using xml on the worl-wide web*. In Smyth, B. and Cunningham, P., editors, Advances in Case-Based Reasoning, volume 1488 of Lecture Notes in Computer Science, pages 274-285. Springer Verlag.
- [Wiederhold92] Wiederhold, G. (1992). *Mediators in the architecture of future information systems*. IEEE Computer, 25(3):38-49.