Designing a Travel Recommendation System using Case-Based Reasoning and Domain Ontology

Camelia Lemnaru, Mioara Dobrin, Mihaela Florea, Rodica Potolea
Technical University of Cluj-Napoca
{Camelia.Lemnaru, Rodica.Potolea}@cs.utcluj.ro, {Mioara.Dobrin, Mihaela.Florea}@student.utcluj.ro

Abstract — **This paper proposes a new travel recommendation system, which aims to provide a unified solution to user demands. It is a hybrid system that combines Case-Based Reasoning with domain-specific ontology. Queries can be constructed template-based or in natural language and a relaxation query management strategy is developed. The solution space is searched from two perspectives: relevance and diversity. We propose a new similarity function, which measures the distances according to the domain ontology.**

Index Terms— **travel recommendation system, case-based reasoning, domain ontology, similarity computation, query relaxation.**

I. Introduction

Tourism has become the world's largest trade, with its expansion still exhibiting a constant year-to-year growth. The Internet has been the major source of tourist destination information for travelers for years. In consequence, a lot of effort has been invested recently to develop applications which support the tourist in selecting travel destinations or related products and services (e.g. attractions, hotels, flights).

This wide spectrum of information is currently provided by multiple sources belonging to different categories: on-line travel agencies, tour operators, cruise operators, airlines, hotel chains. However, extracting information from these heterogeneous sources such as to provide coherent recommendations is a difficult and computationally-intensive task. As the number of query search results is usually highly large, it is not easily manageable by the human reader. Moreover, a destination is a complex concept; it aggregates several elementary components, such as attractions, activities or accommodation facilities, which should convey more information than the simple location or room amenities. Modeling this concept and the decision process in the tourist perspective is still an open and challenging research problem.

We have developed a travel recommendation system, based on a combination of Case-Based Reasoning (CBR) and domain ontology. The system supports the selection of travel related products such as hotel, flight, points of interests and activities, providing a unified solution from which we build a travel plan according to the user preferences.

The rest of the paper is organized as follows: the next sub-section discusses previous work on recommender systems which use case-based reasoning methodologies. The second section presents a detailed description of our approach and its implementation while in section 4 we present the evaluation of important elements of the developed system. The last section concludes the paper.

A. Related work

A series of approaches have been developed in the attempt to provide efficient recommendations by means of an underlying CBR-based architecture. The most prominent are: DieToRecs, Trip@dvice, Bulchino.

DieToRecs is a Case-Based travel planning recommender system, which helps the user plan a leisure travel to a selected destination [1]. Three different recommendation techniques are implemented in DieToRecs: the single item recommendation (SIR), the travel completion (TC) and seeking for inspiration (SI). In DieToRecs, a case represents a user interaction with the system and it is built incrementally during the recommendation session [2].

Trip@dvice is a hybrid recommendation system that integrates CBR, interactive query management and collaborative- based filtering. Trip@dvice bases its recommendations on a case model that capture a unique human–machine interaction session. A case collects information provided by the user during the session, the products selected and some stable user related preferences and demographic data if it is registered [3].

BULCHINO (Bulgarian Cultural Historical and Natural Objects) is a web based catalogue for electronic representation of the Bulgarian culture-historical heritage. The proposed architecture is based on the realization of content based retrieval of cultural objects by means of metadata characterizations and domain ontology inclusion. It implies to use ontology as vocabulary to define complex, multi-relational case structures to support the CBR processes [4].

II. Proposed Solution

Unlike other systems developed using the CBR paradigm, we employ ontologies as persistence media and apply the Description Logics (DL) reasoning process to perform retrieval and similarity computations. In our system a case is represented by two main components: TravelDescription and TravelSolution. Each case component has a set of elements,

978-1-4673-2952-1/12/$31.00 ©2012 IEEE

called attributes from now on. Furthermore, the case structure proposed may contain composite attributes, represented as sets of ontology concepts. We propose new similarity metrics for these composite attributes. Another novelty of our system is the ability to query CBR in natural language. This functionality uses statistical retrieval based on a combination of VSM (Vector Space Model) and Boolean model to determine how relevant a given document is to a query. Our system also contains an Intelligent Query Manager (IQM) module which is able to relax the user query in case the query is very restrictive and the system can't provide any results. The final solution suggested to the user represents an optimal combination of single product items, meaning that at user request we can offer flights related to a preferred travel destination.

A. System Architecture

The architecture of the system is presented in Figure 1. The system supports two main flows: (1) products recommendations based on a user selection from a range of predefined choices or (2) based on the natural language query. The first is designed for a user with specific needs and preferences that fit the query template and the second for a user with a query that does not match the template. Therefore, the user can interrogate the system in two different ways: either by formulating a Template based Query (TQ) or a Natural Language Query (NLQ). TQ manages a query based on a user selection from a set of choices. NLQ processes a query written in natural language. When we refer to the sub-components of a query from the point of view of ontology, we call the query sub-components, atomic queries. The two flows are presented in detail in the following sub-sections.

*Template based recommendation***:** The user interacts with the recommender system by asking for recommendations on a product type (e.g. a destination, flight, attractions, activities) following a template query [5]. The system replies to the query either by recommending products (e.g. hotels, activities, flights) or, in case the query fails (empty result set), by suggesting some relaxed queries returned by the IQM.

Recommendation based on a user query written in natural language: Based on a user input query expressed in natural language (specifying a destination and/or other travel preferences) a search in CBR is performed and the top k most similar cases with the current one are retrieved. The method relies on a combination of VSM and the Boolean model [6] to determine the relevance of a given document to a query. The idea behind the VSM is the correlation between the frequency of a query term in a document relative to the frequency of the term in all the documents in the collection, and the relevance of that document to the query [5].

B. Modules Overview

The system was designed as a component-based architecture and consists of several modules that interact with each other: Query Processors, IQM, Ranking Module, Similarity Module, Textual Statistical Retrieval Module, Aggregated Solution Module.

The Query Processor Module 1 is the component which manages the template-based user request. The atomic components of the query are extracted and interpreted as constraints and a new case is built with an empty TravelSolution and the TravelDescription set according to the query constraints. Based on the current case, a request is sent to the products DB (which contains all available travel products).

The purpose of the IQM Module is to deal with the case of a failing query. It proposes an alternative solution to the user, using several complex queries reformulation strategies of the initial TravelDescription of the query.

When the result set contains a small set of travel solutions, then these are passed to the Case Base Reasoning component, where each travel product is scored in the Ranking Module.

The Similarity Module employs two types of similarity metrics, the ones which are applied on the ontology and the ones are not. Based on this metrics it computes a final score for each product in the Ranking Module. Finally, the ranked products are returned to the user.

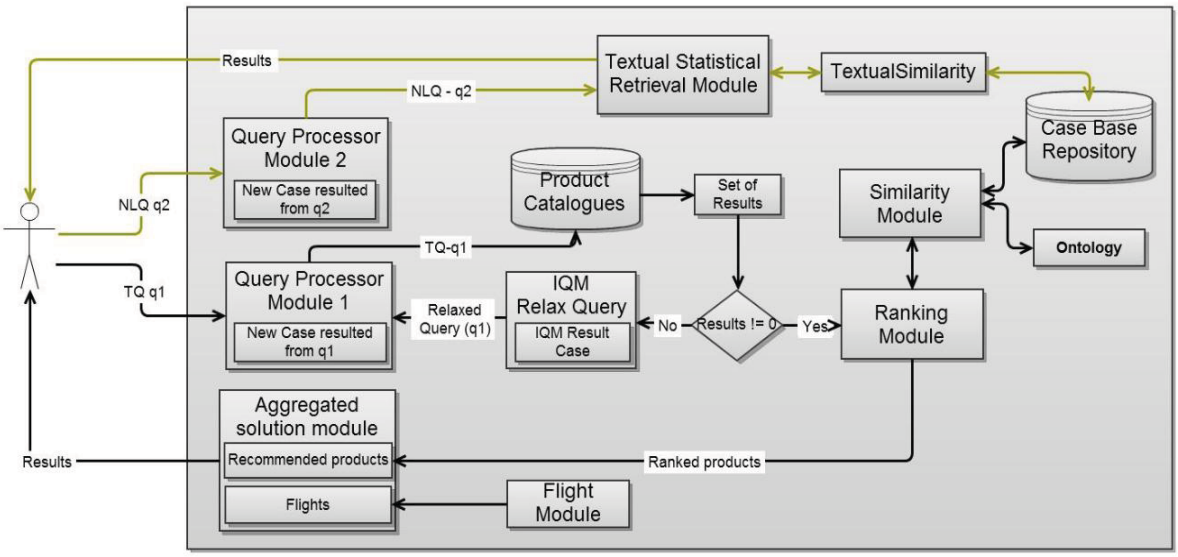

Figure 1: System architecture

The flight module is the component which implements all the required procedures in order to recommend requested flights. It employs an external tool to process user requests and dynamically retrieve data from the internet upon each new request.

The Query Processor Module 2 is a component which manages the user request when it is written in NLQ. From the current query we extract the features we need in order to build a new case. Based on the current case, a request is sent to the CBR and the most similar cases are retrieved.

The Textual Statistical Retrieval Module uses the Boolean model to first narrow down the documents that need to be scored based on the use of Boolean logic in the query specification. There are also added some capabilities and refinements onto this model to support Boolean and fuzzy searching, but essentially the module remains a VSM based system.

C. Case Model

In our system, a case is built incrementally during the recommendation session. A case comprises the following main components (Figure 2): TravelDescription (query constraints) and the corresponding TravelSolution. The user's request represents the query from which we build our Travel Description. Both components describe the collaborative and content features of a case model [5]. The collaborative features capture preferences relevant to the users' decision-making process which cannot generally be mapped into the features of products in the electronic catalogues. The content features are built by querying the descriptors of the products listed in the catalogues. The main difference between TravelSolution and TravelDescription resides in the attribute HotelInfo, which only appears in the TravelSolution.

Table 1 presents the case description attributes (features) with their associated similarity function and a weight according to the attribute relevance in the case. The feature weight values have been determined as a result of evaluations with different combinations of values. A similarity score is computed for each feature, according to its attached function. The global similarity score is aggregated as a weighted average. Another classification criterion for the attributes is represented by their mapping to ontology instances. Those which are mapped onto an ontology are called instance-type attributes (e.g. seasons, Activity, Attraction, Facility) and the others are simple attributes (e.g. TravelType, number of persons, location, transportation mean, duration).

Table 1: TravelDescription attributes

Attribute	Weight	Function
TravelType	1	Equal
NoOfPersons	1	Equal
Location	2	LocationDistance
Transportation	1	Equal
Duration	1	Equal
Seasons	1	Cosine
Accommodation	1	AccommodationDistance
Activity	1	OntoSetActivity
Attraction	1	OntoSetAttraction
Facility	1	OntoSetFacility

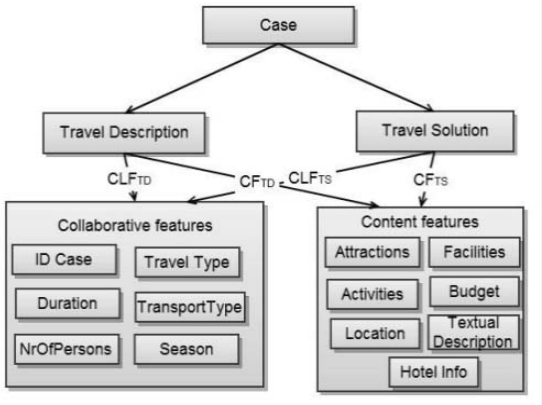

Figure 2: Case structure

The instance-type attributes Activity, Attraction, Facility are compound attributes consisting of sets of ontology instances. Similar to a TravelDescription, the attributes of a TravelSolution have associated similarity metrics to compute the similarity between two solutions.

D. Domain Ontology

To evaluate de similarity between solutions (components) an ontology-based approach has been considered. The ontology allows for identifying the relationships between concepts, broadening or narrowing the search space according to users' needs and context dependencies. Entities of the ontology are components of atomic queries. So, along with their instances and properties allow for evaluating the similarity between entities.

The CBR uses the ontology to measure the semantic distance between similar cases or situations. In the ontology we have concepts which have high relevance in the similarity assessment. An ontology driven similarity computation between concepts has higher relevance because we do not only take into consideration their place in the hierarchy, but also their associated properties. A snapshot of a small part of our ontology is presented in Figure 3, showing the concept Activity, its sub-concepts (e.g. Sport, Relaxation, Adventure) and instances (e.g. Swimming, Golf) are parts of template-based queries.

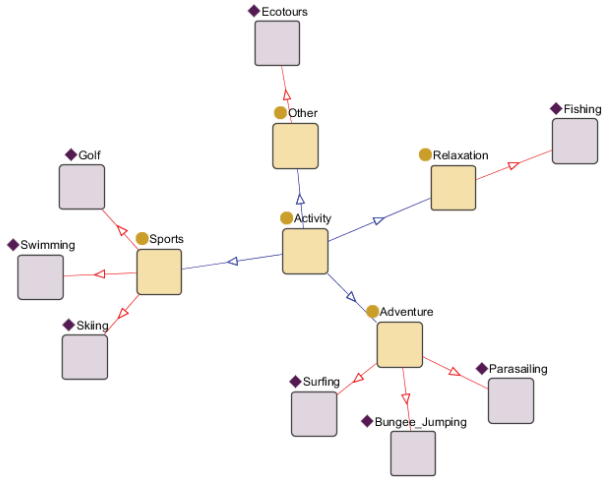

Figure 3: Ontology overview

E. Similarity Computation

Similarity metrics: The similarity metrics aim to assess the computation similarity between the query and a case, and are used to choose the most similar case to a given query. We employ two types of similarity metrics: local (compute the similarity between the attributes) and global (compute the weighted average over the local similarities). We consider two types of attributes on which the similarity metrics are applied. First, the attributes which are represented as ontological concepts, such as Activity, Attraction, and Facility, further referred to as instance-type attributes, for which similarity metrics based on ontology have been considered. Second, attributes that are not mapped onto ontological concepts, but help us compute the similarity between two cases (e.g. location, number of persons, mean of transportation).

Local similarity functions: For each attribute, a local similarity measure determines the similarity between the two attribute values. Each attribute has associated a local similarity function depending on its type. The function has a certain weight according to the relevance of the attribute in the case.
To determine the similarity between simple attributes we have used the following functions: *Equal*, *LocationDistance* and *AccommodationDistance*. The *Equal* function verifies if the values of two attributes are equal, returning 1 for equal, 0 otherwise. *AccommodatonDistance* computes the similarity between two attributes as a cyclic distance and is applied to the stars of a hotel .

LocationDistance function computes the local similarity function between two cities according to their coordinates. First a distance[1] is computed according to the formula (1) and then the value is normalized. Let $\emptyset_S, \lambda_S, \emptyset_f, \lambda_f$ be the geographical latitude and longitude of two points (a base "standpoint" and a destination "forepoint"), respectively, and $\Delta\phi$,$\Delta\lambda$ their differences; then $\Delta\hat{\sigma}$, the central angle between them, is given by the spherical law of cosines [1]:

$$\Delta\hat{\sigma} = \arccos\left(\sin\phi_S \sin\phi_f \cos\phi_S \cos\phi_f \cos\Delta\lambda\right) \quad (1)$$

The distance *d*, i.e. the arc length, for a sphere of radius *r* and $\Delta\hat{\sigma}$ given in radians, is then

$$d = r\Delta\hat{\sigma} \quad (2)$$

For instance-type attributes, numerical similarity functions based on ontologies are used. In order to determine the similarity between two instance-type attributes, we propose the OntProperty similarity metric based on ontology instances.

The instances of the ontology are grouped, each being attached to the related concept. For example, for the instances Golf and Tennis the corresponding super-concept is Sports. Sports is a sub-concept of Activity, which has also Adventure and Relaxation as sub-concepts. Each instance is represented by a set of properties, which are defined according to the concept they are related to. When defining the properties of a certain concept, the most relevant factors have been considered (e.g. for Activity: the environment, the age restriction, the type).

For the instance-type attributes, the OntProperty function computes the similarity between two instances based on the properties of each one. Each property has an attached weight according to its importance in the similarity assessment.

[1] http://en.wikipedia.org/wiki/Great-circle_distance

Table 2: Ontology instance Skiing–property values weights

Property	Value	Type	Weight
hasActivityName	"sport"	String	1
hasAgeRestriction	"none"	String	1
hasEnvironmentType	"mountain"	String	1
hasName	"Skiing"	String	0.1
hasPartner	"false"	boolean	0.1
rdftype	Sports	owlClass	-

One example which illustrates how properties are defined for a certain instance (e.g. instance Skiing of concept Activity), and the corresponding property weights, are presented in Table 2. The weight values have been determined experimentally. Given two instances i1 and i2, the similarity function OntProperty(i1, i2) between them is calculated [5]:

$$\text{OntProperty}(i1, i2) = \frac{\sum_{i=1}^{m}\left(\text{Fs}\left(f_{i1}^{i}, f_{i2}^{i}\right) * \text{Wi}\right)}{\sum_{i=1}^{m}(\text{Wi})} \quad (3)$$

For example, for instances Swimming and Skiing, similarity(Swimming,Skiing) = 0.375, Swimming having the following property values: Swimming = ("sport", "none", "water", "Swimming", "false").

The TDs attributes Facilities, Attraction and Activities represents instance-type attributes which are composed from a set of instances. For these, we have extended the OntProperty function to compute the similarity for the entire set:

$$\frac{\sum_{i\in Q}^{size(Q)} \sum_{j\in C}^{size(C)} Max(OntProperty(Qi, Cj))}{size(Q)} \quad (4)$$

In order to compute the similarity between a case and a query on these composed attributes (Facilities, Attraction and Activities), equation (4) has been applied.

For each instance of the set corresponding to the query, Iquery_i , a Cartesian product is computed between Iquery_i and the corresponding case set. On the resulted set the similarity is computed on each element, which contains a pair of simple instance attributes, from which the maximum value is stored. For the entire query set the final similarity value is computed as an average between all the stored maximums for each instance. For this computation method we have two particular situations: (1) set similarity to 1 in case the query set is empty, (2) set similarity to 0 whenever the case set is empty.

Global similarity functions: The global similarity function computes a weighted average over the local similarity values for each attribute [5]. In table 3 the global similarity between a case and a query is computed from the values for the local similarity between each attribute and their associated weights (GlobalSim(Case,Query) = 0.46).

Table 3: Global similarity computation example

Attribute	Case	Query	Local Sim.	Weights
TravelType	Skiing	Relaxation	0	1
NoOfPersons	4	2	0	1
Location	Sibiu	Cluj	0.82	2
Transportation	car	car	1	1
Duration	5	7	0	1
Seasons	December	May	0.5	1
Accommodation	FourStars	ThreeStars	0.81	1
Activity	[]	[Yoga]	0	1
Attraction	[Buildings]	[Museum]	0.6	1
Facility	[]	[]	1	1

F. Query Reformulation Method

The query reformulation method in the IQM module is aimed at helping the user redefine a query which returns an empty set, or too few results. The purpose of this module is to relax the minimum number of constraints in the initial query in order to reach more results.

A query is obtained by the conjunction of atomic constraints imposed over products. The relaxation method uses the notion of abstraction of the hierarchy of the selected features [7]. The search space is divided into not disjoint sub-spaces $P = \{P_1, P_2, \ldots, P_k\}$, each P_i being responsible for answering to atomic queries. For space items, we consider a collection of feature abstraction hierarchies, $FAH = \{F_1, ..., F_k\}$ where each $F_i = \{f_1, f_2, ..., f_n\}$ is a hierarchy of abstractions, a comparable set of features. For each partition, the system computes a hierarchy of abstractions, among the features of the partition space.

The product space is defined by a collection of features organized in four hierarchies: FAH = {GeneralCriteria, Activities, Attractions, Facilities}, where GeneralCriteria contains features like price, hotelStars, noOfPersons, location etc. The Activities hierarchy contains specific features like Surfing, Skiing etc., the Attraction hierarchy contains features like Museum, Theatre and the Facilities hierarchy contains Internet, TV, Bar, etc.

A complex query Q for our ontology is a conjunction of at most 4 atomic queries $Q = \{P_1 \cap P_2 \cap P_3 \cap P_4\}$; each P_i belongs to a certain hierarchy and imposes constraints over products. P_1 is determined by the Activities hierarchy, P_2 by the Facilities hierarchy, P_3 by the Attraction hierarchy and P_4 by GeneralCriteria hierarchy. Let's take P_1 = {Surfing, Skiing, Aerobics, Parasailing, Ecotours}, P_2 = {Internet, TV, Bar, Breakfast}, P_3 = {Museum}, P_4 = {price=4, stars=4, location= Bucharest}. $Card(P_1) = card(P_2) = 400$, $card(P_3) = 340$, $card(P_4) = 0$. The system removes a constraint from P_1 and one from P_2 and returns a single query having two constraints relaxed.

One way for implementing a hierarchy for a P_i is to sort the similarity of each element with the rest of the elements contained by it. As proposed in [8], the system determines the best attainable match, as in similarity-base retrieval. For each element of the set, a similarity degree is computed as the sum of its similarity with the other elements. The element which is the most dissimilar is considered the element with the lowest abstraction level and is relaxed. Thus, most similar elements are clustered. This method is influenced by how similarities are computed and aims to eliminate the constraints that do not belong to the same category or a specific meaning. In the attempt to sort the partition in descending order, the conclusion was that in many cases a bigger number of constraints are removed than in the case of ascending order.

An alternative to implement a feature hierarchy is to check the total number of items from the database that meet each constraint. The element with the lowest level of abstraction is considered the item that has the lowest number of appearances in the database. As a consequence, the system eliminates a smaller number of constraints.

G. Result Ranking Method

If the result set contains a reasonable (small) set of items (Travel Products), they are passed to the CBR component, where each item is scored in the ranking module. An item gets a higher score if it has a high similarity with other cases. The similarity of two sessions is computed relying on the whole session description. In order to compute a ranking score for a selected travel product, both the TravelDescription and TravelSolution of the referenced cases (top K most similar cases with the current case, retrieved from case base repository) are considered [9].

Although the most similar systems consider the collaborative features only, in the proposed system the similarity-based retrieval exploits all the case content. The main reason for our choice is that all similarity metrics used for concept-type attributes are computed using Ontology functions which are aiming to determine the similarity of the metadata embedded into real products data. After this process, the corresponding TravelSolutions of the retrieved reference cases (reference products) are used to sort the products selected by the user's query (Figure 4).

H. Flight Recommendation

The purpose of this component is to recommend flights according to users' needs and preferences. Flights can be recommended as related products to the ones already suggested, which means that additional constraints over the search space are imposed. This way the user is not forced to complete a template in order to find an appropriate flight, because the needed information is retrieved while recommending other products. Alternatively, the module may answer to specific user requests.

The additional constraints (provided by the user or by the system) are processed in an xml configuration file which is then sent to the WebHarvester[2] tool, which dynamically retrieves data from the internet at each new request (Figure 5).

Flight choices include: destination city/airport, departure city/airport, time interval, departure and return date, with/without stops, airline and class. The component returns results only if certain choices are provided, such as departure and return dates.

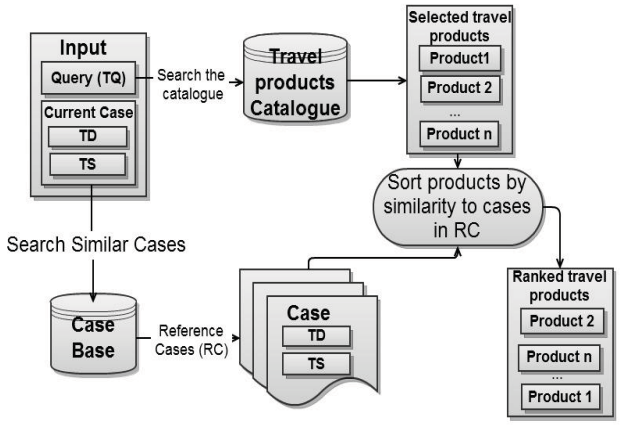

Figure 4: Ranking algorithm

[2] http://web-harvest.sourceforge.net/

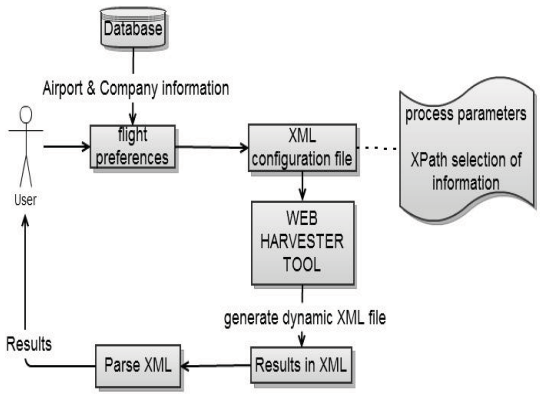

Figure 5: Flights selection process

III. EVALUATION

In order to test and evaluate the recommendation system, we created a relational database which contains all information associated to a travel product. The needed information to perform the tests includes hotel info, attractions and activities related to certain area, data that we gathered from www.hotels.com. Our tests are restricted to data regarding Romanian hotels.

A. Evaluating the similarity functions

Figures 6-8 record the similarity between instances belonging to different ontology concepts; in each figure, the similarity is computed using all available functions: OntDetail, OntCosine, OntDeep and our OntProperty. As indicated by the charts, the other three similarity functions considered yield similar scores, since they only use the instance position in the ontology. The OntProperty function outputs a better separation: e.g. the similarity Skiing-Snowboarding is higher than the similarity Skiing-Swimming, which is rather natural (Figure 6).

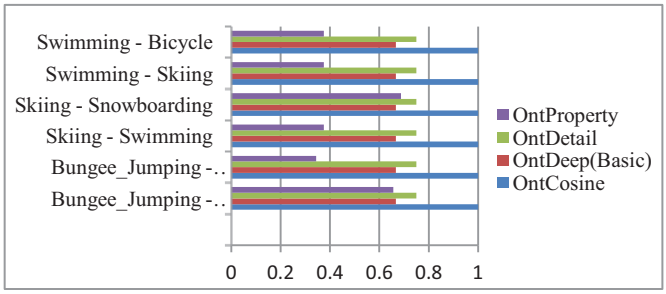

Figure 6: The comparison of different similarity metrics for several pairs of instances (C1,C2) of concept Activity

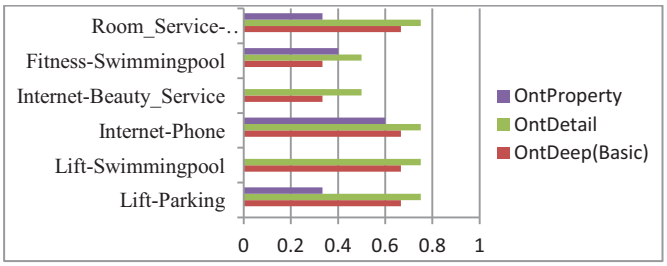

Figure 7: The comparison of different similarity metrics for several pairs of instances (C1,C2) of concept Facility

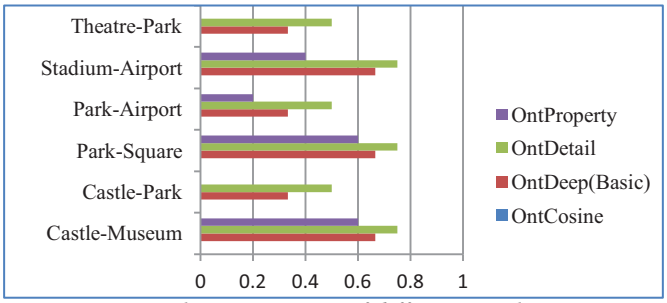

Figure 8: The comparison of different similarity metrics for several pairs of instances (C1,C2) of concept Attraction

B. Evaluating the greedy selection algorithm for case diversity

The list of similar cases retrieved from the CB can be chosen either by selecting the top k most similar cases, or by using a greedy selection algorithm. The algorithm solves the diversity problem, by selecting the cases with the highest combination of similarity to the original query and diversity relative to the re-ranked result list.

In Figure 10 shows the cases selected from the CB in descending order, by similarity. In Figure 9, the cases are selected based on the highest combination of similarity and diversity. In terms of diversity we claim that the greedy selection is more appropriate, as it increases the diversity of the results, while maintaining the relevance, covering a larger area of user interests.

C. Evaluating the query relaxation algorithm

We have chosen twenty different failing queries for testing with IQM module. The tests contain 6 to 20 constrains per query. The sum of the total constraints imposed on the products is 240. The tests were repeated for each of the four methods.

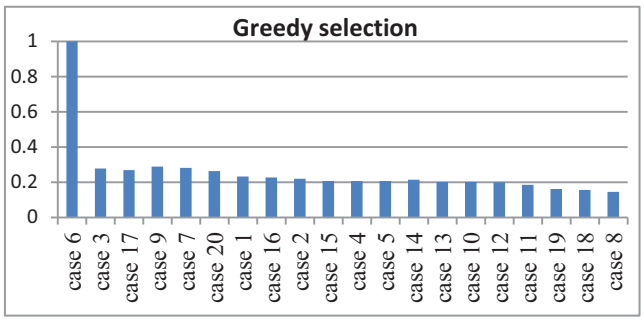

Figure 9: Case ranking using greedy selection for diversity

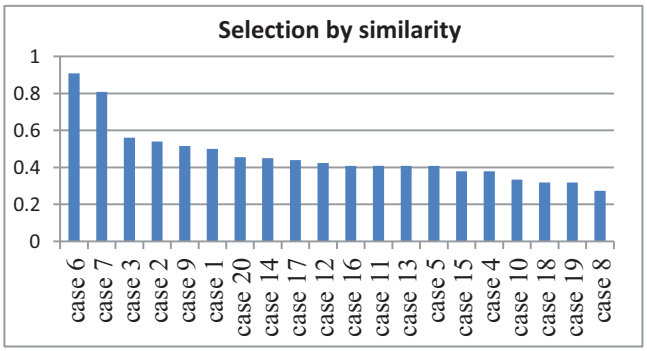

Figure 10: Case ranking using similarity

Reasonable results are reached in time and variety by getting two possible suggestions. The number of constraints relaxed varies with the type of the selected hierarchy; we obtained fewer constraint relaxations when the hierarchy based on frequency has been employed. The time varies depending on the number of constraints added to a query, with lower values for the methods which use the structure with the names of the empty partition. The charts in Figures 11-14 represent the sum of the execution time and the sum of the constraints relaxed for the first choice (least relaxed constraints) in the 20 cases. The following conclusions can be drawn: (1) without structure for empty partitions and hierarchy based on frequency features (Simple Frequency), the number of constraints removed is higher than in the case of using structure for empty partitions, but it eliminates a lower number of constraints than the method which use hierarchy based on its similarity; (2) using the structure for empty partitions and hierarchy based on frequency features yields the lowest number of both relaxed and best time of calculation are obtained; (3) without the structure for empty partitions and hierarchy based on its similarity in structure the highest number of constraints is removed in order to arrive at a result; it also has the highest execution time; (4) using the structure for empty partitions and hierarchy partitions based on similarity the same number of constraints like the previous method are eliminated, but in some cases, in better time.

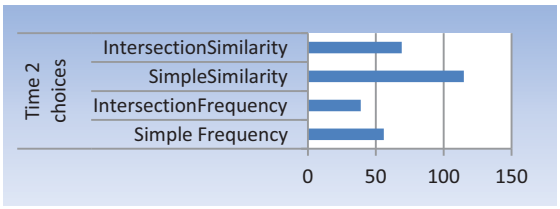

Figure 11: Time taken by the relaxation algorithm, for 2 suggestions

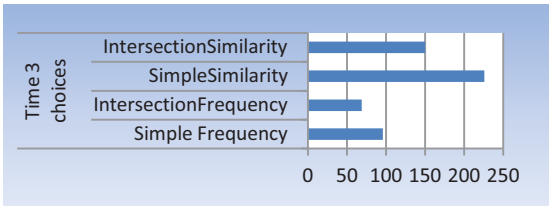

Figure 12: Time taken by the relaxation algorithm, for 3 suggestions

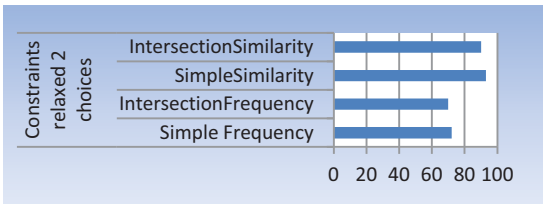

Figure 13: Constraints relaxed by the relaxation algorithm, for 2 suggestions

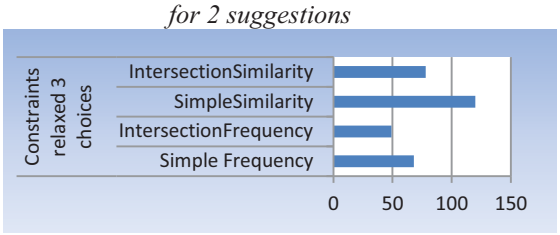

Figure 14: Constraints relaxed by the relaxation algorithm, for 3 suggestions

Thus, both in terms of the number of constraints relaxed and time to attempt the results, the preferred method of relaxation is the one which uses structure for empty partitions and hierarchy on the basis of frequency features.

D. Evaluating the ranking algorithm

The experiments performed for the ranking module attempt to illustrate the way the final product list is reordered according to previous cases which contain similar user preferences to the current case. For these evaluations, an initial TravelDescription (TD) is used as an input and represents the description of the current case illustrated in table 4. Based on this initial case, 3 products were selected from DB. For these 3 retrieved solutions, the ranking algorithm selects 3 most similar cases from CB and computes similarities between their associated TD and TS with each of the 3 solutions. The ranking scores for each selected product:

Product 1: Case 1: 0.6277 * 0.6432 =0.4037; Case 2: 0.5778 * 0.7175 =**0.4145;** Case 3: 0.5454 * 0.6018= 0.3282

Product 2: Case 1: 0.6277 * 0.6353 =0.3987; Case 2: 0.5778 * 0.7308=**0.4222;** Case 3: 0.5454* 0.5775= 0.3149

Product 3: Case 1: 0.6277 *0.6247=0.3921; Case 2: 0.5778 * 0.7030=**0.4061;** Case 3: 0.5454 * 0.6212= 0.3388

The final score of each product is the maximum score over the 2 cases (bold). Therefore, the final products order is: Product 2, Product 1 and Product 3.

Table 4: Similarity computation between the TravelDescription and other 3 cases selected from CB

TravelDescription Attribute	**Travel Description (TD)**	**CB case 1**		**CB case 2**		**CB case 3**	
		Attribute Value	***LS***	***Attribute Value***	***LS***	***Attribute Value***	***LS***
TravelType	Relaxation	Relaxation	1.0	Adventure	0.0	Budget_travel	0.0
NoOfPersons	2	1	0	2	1.0	1	0.0
Location	Cluj-Napoca	Brasov	0.702	Timisoara	0.678	Cluj-Napoca	1.0
Transportation	Train	car	0	car	0	car	0
Duration	7	3	0	7	1.0	1	0
Seasons	August	July	0.99	March	0.49	December	0.49
Accommodation	ThreeStars	Other	0.5	Other	0.5	Other	0.5
Activity	[]	[]	1.0	[Hiking]	1.0	[]	1.0
Attraction	[]	[]	1.0	[Museum, Theatre]	1.0	[]	1.0
Facility	[Internet]	[TV, Internet]	1.0	[]	0.0	[TV, Internet]	1.0
Global Similarity		0.6277		0.5778		0.5454	

IV. Conclusions

Building a complete and efficient recommendation system for the tourism domain poses a series of domain specific challenges: diverse sources of primary data with heterogeneous information; the concept of destination has a complex form, aggregating several atomic components.

This paper proposes a new travel recommendation system, which aims to provide a unified solution to all user demands, i.e. build a travel plan according to the user preferences. It applies Case-Based Reasoning in conjunction with a domain ontology, which enhances similarity computations. Two distinct flows are currently supported by our system: making recommendations based on the user request from a range of predefined choices, or natural language querying. We propose a novel similarity function, OntProperty, for the complex attributes which are instances in the ontology; the new function computes the similarity between such attributes based on their ontology properties.

In order to assess the performance of the strategies proposed in our system, we have performed a series of evaluations, on real-world data which contains hotel info, attractions and activities related to a certain area (restricted to Romania). The results allow the formulation of the following conclusions: (1) the proposed OntProperty similarity function returns more accurate results than the other functions considered, because as it takes into account specific properties for each instance; (2) the greedy case selection algorithm provides an adequate strategy for increasing the diversity of the result set, while maintaining its relevance; (3) the best query relaxation method, in terms of the number of constraints relaxed and time required to provide the results, is the one which employs structure for empty partitions and hierarchy on the basis of frequency features; (4) the ranking algorithm yields a correct order.

Acknowledgement

The IQM module has been implemented by Alexandra Fodor.

References

[1] D. R. Fesenmaier, F. Ricci, E. Schaumlechner, K. W¨ober, and C. Zanella. DIETORECS: Travel advisory for multiple decision styles. In A. J. Frew, M. Hitz, and P. O'Connors – edt, Information and Communication Technologies in Tourism, pp. 232–241. Springer, 2003.

[2] F. Ricci, A. Venturini, D. Cavada, N. Mirzadeh, D. Blaas, and M. Nones. Product recommendation with interactive query management and twofold similarity. In A. Aamodt, D. Bridge, and K. Ashley, edt, ICCBR 2003, the 5th Int. Conf. on Case-Based Reasoning, pp. 479–493, 2003.

[3] A. Venturini and F. Ricci. Applying Trip@dvice Recommendation Technology to www.visiteurope.com, 4th Prestigious Applications of Intelligent Systems (PAIS-2006), The 17th European Conference on Artificial Intelligence, 2006.

[4] N. Govedarova, S. Stoyanov, I. Popchev. An ontology based CBR architecture for knowledge management in BULCHINO catalogue. CompSysTech 2008: 67

[5] M. Dinsoreanu, M. Dobrin, M. Florea, A. Fodor. An integrated Case-based Reasoning and Ontology-driven approach for designing Recommendation Systems. Accepted at the 8th International Conference on Intelligent Computer Communication and Processing, 2012.

[6] K. Saleh, R. Aloufi. Information Retrieval of Text with Diacritics, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.8, August 2010

[7] F. Ricci, N. Mirzadeh and M. Bansal, Supporting user query relaxation in a recommender system, in: *Proceedings of the 5th International Conference in E-Commerce and Web-Technologies - EC-Web*, pp. 31-40, 2004.

[8] D. McSherry. Similarity and compromise. In Aamodt, A., Bridge, D., Ashley, K., eds.: ICCBR 2003, the 5th International Conference on Case-Based Reasoning, Trondheim, Norway (2003) 291–305

[9] F. Ricci, D. Cavada, N. Mirzadeh and A. Venturini, Design of Destination Recommendation Systems. Destination Recommendation Systems: Behavioural Foundations and Applications, pp. 67-94, CAB International 2006