A Logic-Based System for e-Tourism

Francesco Ricca
Università della Calabria
via P. Bucci Cubo 30b, 87036 Rende, Italy
ricca@mat.unical.it

Antonella Dimasi
Exeura Srl
Via Pedro Alvares Cabrai - C.da Lecco 87036 Rende (CS), Italy
antonella.dimasi@exeura.com

Giovanni Grasso
Università della Calabria
via P. Bucci Cubo 30b, 87036 Rende, Italy
grasso@mat.unical.it

Salvatore Maria Ielpa
Università della Calabria
via P. Bucci Cubo 30b, 87036 Rende, Italy
ielpa@dlvsystem.com

Salvatore Iiritano
Exeura Srl
Via Pedro Alvares Cabrai C.da Lecco 87036 Rende (CS), Italy
salvatore.iiritano@exeura.com

Marco Manna, Nicola Leone
Università della Calabria
via P. Bucci Cubo 30b, 87036 Rende, Italy
{manna, leone}@mat.unical.it

Abstract. In this paper we present a successful application of logic programming for e-tourism: the iTravel system. The system exploits two technologies that are based on the state-of-the-art computational logic system DLV: (i) a system for ontology representation and reasoning, called OntoDLV; and, (ii) HiLeX a semantic information-extraction tool. The core of iTravel is an ontology which models the domain of tourism offers. The ontology is automatically populated by extracting the information contained in the tourism leaflets produced by tour operators. A set of specifically devised logic programs is used to reason on the information contained in the ontology for selecting the holiday packages that best fit the customer needs. An intuitive web-based user interface eases the task of interacting with the system for both the customers and the operators of a travel agency.

Keywords: Answer Set Programming, ASP, E-Tourism, Knowledge Representation and Reasoning, Information Extraction

* Address for correspondence: Dipartimento di Matematica, Università della Calabria, via P. Bucci Cubo 30b, 87036 Rende, Italy
1. Introduction

1.1. Scenario and Motivation

In the last few years, the tourism industry has strongly modified marketing strategies with the diffusion of e-tourism portals in the Internet. The efficacy of e-tourism solutions is also witnessed by the continuously growing community of e-buyers who prefer to surf the Internet for buying holiday packages. On the other hand, traditional travel agencies are undergoing a progressive loss of marketing competitiveness. This is mainly due to the presence of web portals, which basically exploit a new market. Indeed, Internet surfers often like to construct their own individual holiday packages by manually searching for flights, accommodation etc. However, the traditional selling process, whose strength lies in both direct contact with customer and knowledge about customer habits, is experiencing a reduced efficiency. This can be explained, to a large extent, by the increased complexity of matching demand and offer. Indeed, travel agencies receive thousands of e-mails per day from tour operators containing new pre-packaged offers. Consequently, the employees of the agency cannot know all the available holiday packages (since they cannot analyze all of them). Moreover, customers are more demanding than in the past (e.g. the classic statement “I like the sea” might be enriched by “I like snorkeling”, or “please find an hotel in Cortina” might be followed by “with beauty and fitness center facilities”) and they often do not declare immediately all their preferences and/or constraints (like, e.g., budget limits, preferred transportation mean or accommodation etc.). The knowledge of customers preferences plays a central role in the traditional selling process. However, the task of matching this information with the large and unstructured e-mail database is difficult and time-consuming to carry out precisely. Consequently, the seller of a travel agency is often unable to find the best possible solution to the customer needs in reasonable time. The iTravel project is motivated by the need to overcome the above-mentioned causes of inefficiency. The main challenges to be faced are the following:

(i) automating the extraction and interpretation of the tourism offers contained in tourism leaflets, which travel agents receive by email,

(ii) automating the population of an easy-to-browse ontology with the incoming tourism offers in (so that they are immediately available for the seller), and

(iii) providing “intelligent” search that combines knowledge about users preferences with geographical information and matches user needs with available offers. Thereby, (i) knowledge on recommended/unrecommended places and dates and (ii) general information such as meteorological and political dangers taken from the internet (e.g., from the web site of the Italian Ministry for Foreigner Affairs), are both taken into account.

1.2. Contributions

We have exploited some tools based on computational logics and, particularly, on Answer Set Programming (ASP) [16] for developing the core functionalities of the iTravel system. ASP is a powerful logic programming language, which is very expressive in a precise mathematical sense: in its general form, allowing for disjunction in rule heads and nonmonotonic negation in rule bodies, ASP can represent every problem in the second level of the polynomial hierarchy [13] in a fully declarative way. The
core functionalities of iTravel are based on two technologies\(^1\) relying on the state-of-the-art ASP system DLV [22]: OntoDLV [32, 33] an ASP-based system for ontology representation and reasoning; and, \(\mathcal{H}L\mathcal{E}X\) [25, 35, 34], a tool for semantic information-extraction from unstructured or semi-structured documents.

In detail, in the iTravel system, behind the web-based user interface (which can be used by both employees agency and customers), there is an “intelligent” core that exploits a rich OntoDLV ontology that models the domain of discourse and the user preferences, and stores all the available data. Geographic information was obtained by including GeoNames [29], one of the largest publicly-available geographical databases, and enriched by modeling the knowledge of the travel agent regarding places and offered holidays. The ontology is automatically populated with tourism offers by extracting the information contained in the tourism leaflets produced by tour operators. It is worth noting that, offers are mostly received by the travel agency in a dedicated e-mail account. Moreover, the received e-mails are human-readable, and the details are often contained in email-attachments of different format (plain text, html, pdf, gif, or jpeg files) containing a mix of text and images. The \(\mathcal{H}L\mathcal{E}X\) system allows for automatically processing the received contents, and for populating the ontology with the data extracted from tourism leaflets. Once the information is loaded onto the ontology, the user can perform an “intelligent” search for selecting the holiday packages that best fit the customer needs. iTravel tries to exploit the knowledge of the typical employee of a travel agency by using a set of specifically devised logic programs that “reason” on the information contained in the ontology.

In the remainder of the paper, we first introduce the employed ASP-based technologies; then, in Section 3, we describe how the crucial tasks have been implemented; we show the architecture of the iTravel system in Section 4; in Section 5 we report on the performance of iTravel on a real-world scenario. in Section 6 we discuss related work; finally, we draw the conclusion in Section 7.

2. Underlying ASP-based Technologies

The core functionalities of the e-tourism systems iTravel are based on two technologies relying on the DLV system [22]: OntoDLV [32, 33] a powerful ASP-based ontology representation and reasoning system; and, \(\mathcal{H}L\mathcal{E}X\) [25, 35, 34], a tool for semantic information-extraction from unstructured or semi-structured documents. In the following we briefly describe both OntoDLV and \(\mathcal{H}L\mathcal{E}X\), the reader interested in a more detailed description is referred to [32, 33] and [25, 35, 34], respectively.

2.1. The OntoDLV System

Traditional ASP in not well-suited to ontology specifications, since it does not directly support features like classes, taxonomies, individuals, etc. Moreover, ASP systems are a long way from comfortably enabling the development of industry-level applications, mainly because they lack important tools for supporting programmers. All the above-mentioned issues were addressed in OntoDLV [32, 33] a system for ontologies specification and reasoning. Indeed, by using OntoDLV, domain experts can create, modify, store, navigate, and query ontologies; and, at the same time, application developers can easily

\(^1\)Both systems are developed by Exeura srl, a technology company working on analytics, data mining, and knowledge management, that is investing on their industrialization finalized to commercial distribution.
develop their own knowledge-based applications on top of OntoDLV, by exploiting a complete Application Programming Interface [14]. OntoDLV implements a powerful logic-based ontology representation language, called OntoDLP, which is an extension of (disjunctive) ASP with all the main ontology constructs including classes, inheritance, relations, and axioms. In OntoDLP, a class can be thought of as a collection of individuals who belong together because they share some features. An individual, or object, is any identifiable entity in the universe of discourse. Objects, also called class instances, are unambiguously identified by their object-identifier (oid) and belong to a class. A class is defined by a name (which is unique) and an ordered list of attributes, identifying the properties of its instances. Each attribute has a name and a type, which is, actually, a class. This allows for the specification of complex objects (objects made of other objects). Classes can be organized in a specialization hierarchy (or data-type taxonomy) using the built-in is-a relation (multiple inheritance). Relationships among objects are represented by means of relations, which, like classes, are defined by a (unique) name and an ordered list of attributes (with name and type). OntoDLP relations are strongly typed while in ASP relations are represented by means of simple flat predicates. Importantly, OntoDLP supports two kind of classes and relations: (base) classes and (base) relations, which correspond to basic facts (that can be stored in a database); and collection classes and intensional relations, which correspond to facts that can be inferred by logic programs; in particular, collection classes are mainly intended for object reclassification (i.e., for repeatedly classifying individuals of an ontology). For instance, the following statement declares a class modeling customers, which has six attributes, namely: firstName, lastName, and status of type string; birthDate of type Date; a positive integer childNumber, and job which contains an instance of class Job.

\[
\text{class Customer (firstName: string, lastName: string, birthDate: Date, status: string, childNumber: positive integer, job: Job).}
\]

As in ASP, logic programs are sets of logic rules and constraints. However, OntoDLP extends the definition of logic atom by introducing class and relation predicates, and complex terms (allowing for a direct access to object properties). This way, the OntoDLP rules merge, in a simple and natural way, the declarative style of logic programming with the navigational style of the object-oriented systems. In addition, logic programs are organized in reasoning modules, to take advantage of the benefits of modular programming. For example, with the following program we single out the pairs of customers having the same birthdate:

\[
\text{module (CustomersWithTheSameBirthDate) { sameBirthDate(C1,C2,D) :- C1: Customer(birthDate:D), C2: Customer(birthDate:D).}}
\]

Classes and relations can be populated by either asserting logic facts or by exploiting the information contained in existing relational databases [5]. In the latter case, the instances are defined by means of mapping rules that “virtually import” the data from a given database. For instance the following specification populates Continent class by taking the information from table continent_codes in the database identified by geo_db:

\[
\text{virtual class Continent ( name:string ) { f(ID):Continent( name:Name ) :- continent_codes@geo_db( geoname_id:ID, name:Name ).}}
\]

Here, the rule acts as mapping between the data contained in table continent_codes and the instances of class continent; whereas functional object identifiers are suitably built from database values for identifying ontology instances. This kind of classes and relations are called virtual because their instances
come from (and might permanently reside in) an external source; but, as far as reasoning and querying are concerned they are like any other class directly specified in OntoDLP.

The core of OntoDLV is a rewriting procedure [33] that translates ontologies, and reasoning modules to an equivalent standard ASP program which, in the general case, runs on the state-of-the-art ASP system DLV [22]. OntoDLV features a persistency manager that allows one to store ontologies transparently both in text files and internal relational databases; while powerful type-checking routines are able to analyze ontology specifications and single out consistency problems. Importantly, if the rewritten program is stratified and non-disjunctive [16, 15, 28] (and the input ontology resides in relational databases) then the evaluation is carried out directly in mass memory by exploiting a specialized version of the same system, called DLV $DB$ [40]. Note that, since class and relation specifications are rewritten into stratified and non-disjunctive programs, queries on ontologies can always be evaluated by exploiting a DBMS. This makes the evaluation process very efficient, and allows the knowledge engineer to formulate queries in a language more expressive than SQL.

2.2. The $HL\_X$ System

$HL\_X$ [25, 35, 34] is a system for ontology-based information extraction from semi-structured and unstructured documents, which has already been employed in many relevant real-world applications. The $HL\_X$ system implements a semantic approach to the information extraction problem based on a new semantic conceptual model by means of: (i) ontologies as knowledge representation formalism; (ii) a general document representation model for unifying different input formats (html, pdf, doc, ...); and, (iii) the definition of a formal attribute grammar able to describe, by means of declarative rules, objects/classes w.r.t. a given ontology. Most of the existing information extraction approaches do not work in a semantical way and they are not independent of the specific type of document they process. Contrariwise, the approach implemented in $HL\_X$ confirms that it is possible to recognize, extract and structure relevant information from heterogeneous sources. $HL\_X$ is based on OntoDLP for describing ontologies, since this language perfectly fits the definition of semantic extraction rules. Regarding the unified document representation, a document (unstructured or semi-structured) is seen as a suitable arrangement of objects in a two-dimensional space. Each object has its own semantics, is characterized by some attributes and is located in a two-dimensional area of the document called portion. A portion is defined as a rectangular area univocally identified by four cartesian coordinates of two opposite vertices. Each portion “contains” one or more objects and an object can be recognized in different portions.

The language of $HL\_X$ is founded on the concept of ontology descriptors. A “descriptor” looks like a production rule in a formal attribute grammar, where syntactic items are replaced by ontology elements, and where extensions for managing two-dimensional objects are added. Each descriptor allows us to describe: (i) an ontology object in order to recognize it in a document; or (ii) how to “generate” a new object that, in turn, may be added in the original ontology. Note that an object may also have more than one descriptor, for recognizing the same kind of information when it is presented in different ways.

3. The iTravel System

In this section we describe the core of the iTravel system: its key features based on ASP. iTravel is an e-tourism system conceived for classifying and driving the search of pre-packaged tourism offers for both travel agencies operators and their customers. The system, like other existing portals, is equipped
with a proper (web-based) user interface; but, behind the user interface there is an “intelligent” core that exploits knowledge representation and reasoning technologies based on ASP. In iTravel (see Figure 1 (a)), the information regarding the tourism offers provided by tour operators is mainly received by the system as a set of e-mails. Each e-mail might contain plain text and/or a set of leaflets, usually distributed as html, pdf or image files which store the details of the offer (e.g., place, accommodation, price etc.). Leaflets are devised to be human-readable, might mix text and images, and usually do not have the same layout. E-mails (and their content) are automatically processed by using the HX system (i.e., e-mails are “unwrapped”: attachments are separately processed, enclosed external links are followed and corresponding web pages analyzed); and, the extracted data about tourism offers is used to populate an OntoDLP ontology that models the domain of discourse: the “tourism ontology”.

Note that, the automatic population of the ontology is one of the key features of the system; but iTravel can deal also with sources different from e-mails. Indeed, the internal ontology can be also manually-populated by travel agents by using a proper form, and additional leaflets can be given in input to the extraction system. The resulting ontology is, then, analyzed by exploiting a set of reasoning modules (ASP programs) combining the extracted data with knowledge about places (geographical information) and user preferences specified in the tourism ontology. The system mimics the typical deductions made by a travel agency employee for providing appropriate answers to the user needs.

It is worth pointing out that the final goal of the system is to provide an effective interface for the employees of a travel agency, and, thus, iTravel is not equipped with a reservation system (travel agencies already own conventional systems for managing reservations). The same holds for the customer interface, which only allows for browsing the holiday packages. Indeed, the customer interface has been devised with the intent of welcoming customers and offering a comfortable access to the travel agency. The goal is also to reduce the “visiting-only” phenomenon, which consists in the repeated visit of customers that not necessarily end in a purchase. In the following subsections, we describe the main components of the tourism ontology and the implementation of the above-mentioned ASP-based features. It is worthwhile noting that, for clarity of presentation, we show a simplified version of both ontology and reasoning modules (logic programs) where noisy details (like, e.g., namespaces) that would have made the specification only longer and distracting were removed.
3.1. The Tourism Ontology

The “tourism ontology” has been developed with the cooperation of the staff of a real travel agency. In this way, we could model the key entities that describe the process of organizing and selling a holiday package. In particular, the tourism ontology models: geographic information, travel agent knowledge, user preferences, and tourism offer information.

Geographical Information. The most relevant ontology entities that model geographical information are reported in Figure 2. World locations are modeled by class Place, and partitioned into six (sub)classes, one for each tourism-relevant kind of place: Continent, Nation, AdminSubArea, PopulatedPlace, Mountain, and Island. In practice, places are hierarchically organized into four levels according to their natural containment relationship. In particular, the basic place-containment is modeled in the ontology by means of five relations, namely: ContinentContainsNation, NationContainsAdminSubArea, AdminSubAreaContainsPopulatedPlace, AdminSubAreaContainsMountain, and AdminSubAreaContainsIsland. The first one models the containment of nations in continents; whereas we considered only one level of subdivision of nations, which was determined in accordance with their administrative organization (e.g., Italy is divided into “regioni”, England is divided into “countries”, etc.); on the last level, there are “populated places” (i.e., cities and villages), mountains and islands, which, in turn, are contained in administrative areas. The complete part-of hierarchy of places, containing all direct and indirect inclusions, is obtained with the intensional relation Contains. Indeed, Contains is defined by means of a logic program made up of six logic rules (see Figure 2): one for each virtual relation modeling basic place-containment, and an additional one that basically encodes a transitive closure. Note that, ASP rules allowed us to define (and compute) the full place-inclusion hierarchy in a simple yet effective way. In the following it will be clear that the information modeled in the Contains relation is crucial for the effectiveness of both extraction process and package search.

The relation AlternateNames associates all the known toponyms with corresponding places (e.g. a different one for each language); this information is exploited during both the extraction process and user queries pre-processing steps, in order to correctly disambiguating the input. The additional TouristicRegion and TouristicRegionContainsPlace modeling tourism regions crossing political borders (e.g. Tyrol-South-Trentino is a cross-border region collecting Austria’s Tyrol region and Italy’s Trentino Alto Adige). This information was manually-inserted in the ontology with the help of the domain experts.

The above-described ontology entities have been populated by exploiting one of the largest publicly-available geographical databases: GeoNames [29]. GeoNames contains thousand of places, alternative toponyms and related geographic information. In order to map the GeoNames database to our ontology we exploited both virtual classes and virtual relations [5]. In particular, some of the mappings between tourism ontology classes and GeoNames tables are reported in Figure 3. The first statement of Figure 3 links the GeoNames database to the ontology by defining the database identifier “geo_db”; whereas, the instances of class Continent (resp. Nation) are obtained from table continent_codes (resp. country_info) by means of a single mapping rule where database tables are referred by exploiting sourced atoms. Note that, object identifiers are univocally built by exploiting the database code geoname_id. Finally, ContinentContainsNation is given by the join of tables continent_codes and country_info. The remaining virtual classes and relations were defined in a similar way.

---

2Note that, apart from class Place and relation Contains, all the remaining (basic) ontology entities in Figure 2 are virtual.
3A description of GeoNames is not reported here for space reasons. The interested reader can find it on the Web at [29].
% places and related information
class Place ( name:string, kind:PlaceKind ).

class PlaceKind ()..
continent:PlaceKind(). nation:PlaceKind(). adminSubArea:PlaceKind().
populatedPlace:PlaceKind(). mountain:PlaceKind(). island:PlaceKind().

% virtual classes taking places from GeoNames
virtual class Continent( name:string, kind:PlaceKind ) isa Place.
virtual class Nation( name:string, kind:PlaceKind ) isa Place.
virtual class AdminSubArea( name:string, kind:PlaceKind ) isa Place.
virtual class PopulatedPlace( name:string, kind:PlaceKind ) isa Place.
virtual class Mountain( name:string, kind:PlaceKind ) isa Place.
virtual class Island( name:string, kind:PlaceKind ) isa Place.

% alternative place-names from GeoNames
virtual relation AlternativeName ( place:Place, alternateName:string ).

% tourism regions crossing political borders
class TouristicRegion isa Place.
relation TouristicRegionContainsPlace ( ctr:TouristicRegion, ctd:Place ).

% place containment
intentional relation Contains ( ctr:Place, ctd:Place ).
{
Contains( ctr:P1, ctd:P2 ) :- ContinentContainsNation( ctr:P1, ctd:P2 ).
Contains( ctr:P1, ctd:P2 ) :- NationContainsAdminSubArea( ctr:P1, ctd:P2 ).
Contains( ctr:P1, ctd:P2 ) :- AdminSubAreaContainsPopulatedPlace( ctr:P1, ctd:P2 ).
Contains( ctr:P1, ctd:P2 ) :- AdminSubAreaContainsMountain( ctr:P1, ctd:P2 ).
Contains( ctr:P1, ctd:P2 ) :- AdminSubAreaContainsIsland( ctr:P1, ctd:P2 ).
Contains( ctr:P1, ctd:P2 ) :- TouristicRegionContainsPlace( ctr:P1, ctd:P2 ).

Contains( ctr:P1, ctd:P2 ) :- Contains( ctr:P1, ctd:P3 ), Contains( ctr:P3, ctd:P2 ).
}

% virtual relations reconstructing basic place containment from GeoNames
virtual relation ContinentContainsNation ( ctr:Continent, contained:Nation ).
virtual relation NationContainsAdminSubArea ( ctr:Nation, contained:AdminSubArea ).
virtual relation AdminSubAreaContainsPopulatedPlace ( ctr:AdminSubArea,
                                      ctd:PopulatedPlace ).
virtual relation AdminSubAreaContainsMountain ( ctr:AdminSubArea, ctd:Mountain ).
virtual relation AdminSubAreaContainsIsland ( ctr:AdminSubArea, ctd:Island ).

Figure 2. Tourism Ontology: Geographic Information.
Travel Agent Knowledge. In the ontology, the mere geographic information is enriched by the knowledge that is usually exploited by travel agency employees for selecting a travel destination (see Figure 4). For example, travel agents know which place offers a specific kind of trip (e.g. Kenya offers safari, Sicily offers both sea and sightseeing). To this end, the tourism ontology contains the class TripKind (e.g. safari, cruise, etc. are instances of the TripKind class) and relation PlaceOffer that associates a kind of trip with places. Nevertheless, a place might offer some specific facilities (e.g. in Marsa Alam Egypt it is possible to go snorkeling). Thus, the ontology contains a class called PlaceFacility and a binary relation PlaceOffersFacility associating facilities with places. Moreover, travel agents might suggest avoiding sea holidays in winter, or going to India during the wet monsoon period; whereas, a visit to Sicily might be recommended during summer. This information is encoded in the ontology by means of the two relations: RecommendedPeriod, BadPeriod, respectively associating places with periods in which they should be visited or avoided. It is also the case that places engaged in a war or subjected to terrorism (like, eg. Afghanistan today) should not be recommended for a holiday; thus, relation DangerousPlace lists places that are currently considered risky.

It is worth noting that the instances of classes and relations modeling the knowledge of travel agents were manually-inserted (in accordance with the information provided by a real travel agent and/or publicly available on the Internet) for a large number of places, and can be updated in the system through the travel agent interface.

Tourism Offer Information. Holiday packages are modeled by the TouristicOffer class (see Figure 4). In detail, for each holiday package we store: both the place of departure and the final destination; the kind of trip; cost and duration (in days); the tour operator selling the package (which is an instance of the TourOperator class); an url pointing to the original leaflet, and the expiration date (which is exploited by the system for periodically removing expired offers). The instances of TouristicOffer are added either au-
% travel agent knowledge
relation PlaceOffer ( place: Place, kind: TripKind ).
relation SuggestedPeriod ( place: Place, period: HolidayPeriod ).
relation BadPeriod ( place: Place, period: HolidayPeriod ).
relation DangerousPlace ( place: Place ).
class PlaceFacility ().
relation PlaceOffersFacility ( place: Place, fac: PlaceFacility ).

% tourism offer information
class TouristicOffer ( start: Place, destination: Place,
                        kind: TripKind, cost: positive integer, fromDay: Date, toDay: Date,
                        period: HolidayPeriod, maxDuration: positive integer,
                        deadline: Date, uri: string, tourOperator: TourOperator ).
class TripKind ().
class HolidayPeriod ().
class TourOperator ( name: string, company: string, phone: string, fax: string,
                     email: string, contact: string ).
class TransportationMean ().
class AccommodationKind ().
class AccommodationFacility ().
relation OfferMean ( offer: TouristicOffer, means: TransportationMean ).
relation OfferAccommodation ( offer: TouristicOffer, acc: AccommodationKind,
                            descr: string ).

% user profile
class Customer ( firstName: string, lastName: string, birthDate: Date, status: string,
                 childNumber: positive integer, job: Job ).
relation CustomerPreferredTrip ( cust: Customer, kind: TripKind ).
relation CustomerPreferredMean ( cust: Customer, mean: TransportationMean,
                               relevance: positive integer ).
relation CustomerPreferredAccommodation ( cust: Customer, acc: AccommodationKind,
                                     relevance: positive integer ).
relation CustomerPreferredAccommodationFacility ( cust: Customer, fac: AccommodationFacility,
                                                  relevance: positive integer ).
relation CustomerPreferredPlaceFacility ( cust: Customer, fac: PlaceFacility,
                                         relevance: positive integer ).

Figure 4. Tourism Ontology: Travel Agent Knowledge, Offers Information and User Profile.

tomatically, by exploiting the H\textDelta\textSigma system (see next Section), or manually (via the user interface) by the personnel of the agency. Optionally, tourism offers might have associated some transportation means,\(^4\)

\(^4\)In case there is more than one alternative transportation mean, the ontology is intended to be filled with the main one.
an accommodation kind (e.g. hotel, apartment, etc.) and some accommodation facilities (e.g. swimming pool, spa service, etc.). This additional information is modeled in the ontology by means of the following classes: *TransportationMean*, *AccommodationKind* and *AccommodationFacility*. The association of this information to tourism offers is modeled by the following relations: *OfferMean*, *OfferAccommodation* and *OfferAccommodationFacility*.

**User Profile.** In order to personalize the trip-search, user profiles are also modeled in the tourism ontology. The class *Customer* models the personal information of each customer. User preferences are modeled by exploiting a number of relations, namely: *CustomerPreferredTrip*, *CustomerPreferredMean*, *CustomerPreferredAccommodation*, *CustomerPreferredAccommodationFacility*, and *CustomerPreferredPlaceFacility*. The first relation associates a preferred trip kind to each customer; whereas, the remaining ones associate, for each customer, a preference score with each specific facility that might be associated to an offer, namely: transportation means, accommodation kind, accommodation facilities and place facilities. It is worth pointing out that, at the moment, the user profile data is manually-inserted in the ontology and can be updated by either the travel agent or the customer himself.

### 3.2. Automatic Extraction of Tourism Offers

Tourism offers are mainly available in digital format and they are received via e-mail. It is usual that more than a hundred emails per day crowd the mail folder of a travel agency, and often the personnel cannot even analyze the entire in-box. This causes a loss of efficiency in the selling process, because some interesting offers may be ignored. Note that, most of the information is contained in html, pdf, gif or jpeg files attached to the e-mail messages, and this strongly limits the efficacy of standard search tools like, e.g., the ones provided by e-mail clients.

To deal with this problem, the iTriple system is equipped with an automatic text extraction system based on $\mathcal{HLX}$. Basically, after some pre-processing steps, in which e-mails are automatically read from the inbox, and their attachments are properly handled (e.g. image files are analyzed by using OCR software), the input is sent to $\mathcal{HLX}$. In turn, $\mathcal{HLX}$ is able to both extract the information contained in the e-mails and populate the *TouristicOffer* class. This was obtained by encoding several ontology descriptors (actually, we provided several descriptors for each kind of file received by the agency). For instance, the following descriptor:

```xml
<TouristicOffer (Destination, Period)> -> <X:place(XX){Destination:=X;}>
<X:date(XX){Period:=X;} SEPBY <X:separator()>.```

extracts from the leaflet in Figure 5(a) that the proposed holiday package regards trips to both the Caribbean islands and Brazil. Moreover, it also extracts the period in which this trip is offered. The extracted portions are outlined in Figure 5(b). The result of the application of this descriptor are two new instances of the TouristicOffer class.

### 3.3. Personalized Trip Search

The second crucial task carried out by the iTriple system is the personalized trip search. This feature has been conceived to make simpler the task of selecting the holiday packages that best fit the customer
needs. To this end, we “simulate” the deductions made by an employee of the travel agency in the selling process by using a set of reasoning modules, i.e. a set of specifically devised logic programs.

In a typical scenario, when a customer enters the travel agency, an employee tries to understand his current desires and preferences at first; then, the seller begins to match the obtained information with a number of pre-packaged offers. Actually, a number of candidate offers are proposed to the customer and, then, interpreting his preferences, the employee is able to understand the customer needs. Customer preferences depend on his personal information but also on his holiday habits. Actually, while the seller usually elicits such information by interviewing the client, in case of regular customer he may already have the necessary knowledge. In this process, what has to be clearly understood (for properly selecting a holiday package fitting the customer needs) is summarized in the following four words: where, when, how, and budget. Indeed, the seller has to understand where the customer desires to go; when he can/wishes to leave; how much time he will dedicate to his holiday; which is the preferred transportation means (how); and, finally, the available budget. However, the customer does not directly specify all this information, for example, he can ask for a sea holiday in January but he does not specify a precise place, or he can ask for a kind of trip that is impracticable in a given period (e.g. winter holiday in Sicily in August). In this case the seller has to exploit his knowledge of the world for selecting the right destination and a good offer in a huge range of proposals. This is exactly what the iTravel system does while searching for a holiday package. Current needs are specified by filling an appropriate search form, where some of the key information has to be provided (i.e. where and/or when and/or available money and/or how). Note that, the tourism ontology models both the knowledge of the seller and the profile of customers; moreover, the extraction process continuously populates the ontology with new tourism offers and periodically removes the expired ones. Thus, the system, by running some specifically devised reasoning modules, combines the specified information with the one available in the ontology, and shows the holiday packages that best fit the customer needs.

The most relevant reasoning modules exploited by the iTravel system are reported in Figure 6. The first module, named match profile, allows for ranking the available offers w.r.t. the user preferences. In particular, the first four rules single out the scores obtained by offers w.r.t. the profile of customers. In detail, the first rule can be read as follows: “for each customer \(C\) associate to each offer \(O\) the score \(R\) if the transportation mean included in \(O\) has relevance \(R\) according to the preferences of \(C\)”. In a similar way, the next three rules associate, for each customer, a specific score with available offers depending on: accommodation kind, accommodation facilities and destination facilities, respectively. Finally, the auxiliary predicate preferenceSatisfaction computes the overall satisfaction ranking obtained by available offers. Given a customer \(C\), the overall satisfaction ranking \(Sat\) for an offer \(O\) is obtained
by summing up all the scores obtained by the features included in $O$ (in accordance with the preferences of $C$). The ranking defined by $\text{preferenceSatisfaction}$ is, then, exploited by the system either for (a) proposing some possibly interesting offer to the user when he/she logs-in; or (b) for ranking the results obtained during the search of tourism offers. Task (a) is implemented by evaluating module $\text{welcome recommendations}$; whereas, task (b), is encoded in the module called $\text{personalized search}$ (see Figure 6). Basically, module $\text{welcome recommendations}$ proposes to an incoming old customer some “tempting” offer. More in detail, predicate $\text{loginOffers}$ is defined that lists the tourism offers that propose the trip kind preferred by $C$ and maximize the satisfaction of user preferences. However, the customer might have in mind some specific holiday, and in this case, the system has to search a holiday package that matches the current customer requests. Suppose that the customer specifies destination and period, then the $\text{personalized search}$ module creates a suitable selection of holiday packages as follows. The first two rules select $\text{requested places}$ by means of $\text{Contains}$ relation (i.e. we consider the specified destination together with all the places it contains). Note that, this additional information plays a fundamental role for improving the search; indeed, it is common that the user specifies a nation (e.g., “Italy”) and the system has to consider also packages having a specific place (e.g., “Rome”) as destination. Then, rule (iii) singles out each offer $O$ having a requested place as destination. However, it might be the case that a matching offer is not recommendable (and we call it “bad offer”) either because the chosen period is not the right one for the specified destination (rule iv) or because a trip to the specified destination is currently considered risky (rule v). If some “bad offer” matches the customer query, then the system prompts an alert message next to the search results; moreover, the system might recommend some alternative for the specified period which is not bad (rule vi). Note that, predicates collecting both matching and alternative holiday packages associate to found offers a preference satisfaction score. In this way, results can be ordered by the user interface according to user preferences and offers that best fit the customer needs are showed first. Several other search criteria can be specified, including trip kind, tour operator, and so on. Actually, depending on the input parameters, the system selects a specific reasoning module obtained by suitably modifying the one described above. For instance, when the user specifies both trip kind and holiday period rule (i) is replaced by:

\[
\text{requestedPlace}(P) :\leftarrow \text{query}(\text{tripKind}:K), \text{PlaceOffer}(\text{place}:P, \text{kind}:K).
\]

where requested places are the ones offering the specified trip kind. Otherwise, when the budget is also specified both rule (iii) and rule (iv) are replaced by:

\[
\text{matchingOffer}(O) :\leftarrow O:\text{TouristicOffer}(\text{destination}:P, \text{cost}:C), \text{requestedPlace}(P), \text{query}(\text{budget}:B), C \leq B.
\]

\[
\text{recommendedAlternative}(\text{Off},\text{Sat}) :\leftarrow \text{query}(\text{customer}:\text{Cust}, \text{period}:\text{Per}, \text{budget}:\text{B}), \text{SuggestedPeriod}(\text{place}:\text{Dest}, \text{period}:\text{Per}), \text{not} \text{DangerousPlace}(\text{place}:\text{Dest}), \text{Off}:\text{TouristicOffer}(\text{destination}:\text{Dest}, \text{cost}:\text{C}), \text{preferenceSatisfaction}(\text{Cust}, \text{Off}, \text{Sat}), C \leq B.
\]

where the additional condition is considered in the body and so on.

Remark. The above-reported logic programs are a ready-to-execute declarative specification of the reasoning tasks performed by iTriavel. This specification is run (see Section 4) by exploiting the OntoDLV system. Similar considerations can be done for the Hilex descriptors employed in the extraction tasks.
% Offer ranking based on customer preferences

module(match profile) {
  CustomerPreferredMean( cust:C, mean:M, relevance:R ),
  satisfiesCustomerPreference( O, C, R ) :- O:TouristicOffer(),
  CustomerPreferredAccomodation( cust:C, acc:A, relevance:R ),
  OfferAccomodation( offer:O, acc:A ).
  satisfiesCustomerPreference( O, C, R ) :- O:TouristicOffer(),
  CustomerPreferredAccomodationFacility( cust:C, fac:AF, relevance:R ),
  OfferAccomodationFacility( offer:O, fac:AF ).
  satisfiesCustomerPreference( O, C, R ) :- O:TouristicOffer( destination:P ),
  CustomerPreferredPlaceFacility( cust:C, fac:PF, relevance:R ),
  PlaceOffersFacility( place:P, fac:PF ).

  preferenceSatisfaction( C, O, Sat ) :- C:Customer(), O:TouristicOffer(),
  #sum{ R : satisfiesCustomerPreference( O, C, R ) } = Sat. }

% Recommend best offers when the user logs-in

module(welcome recommendations) {
  loginOffer( C, O ) :- query( customer:C ), CustomerPreferredTrip( cust:C, kind:K ),
  O:TouristicOffer( kind:K ), preferenceSatisfaction( C, O, MaxSat ),
  MaxSat = #max{ Sat : preferenceSatisfaction( C, O, Sat )}. }

% Search offers - basic method

module(personalized search) {
  % select available offers matching query
  (i) requestedPlace( P ) :- query( place:P ).
  (ii) requestedPlace( Pc ) :- requestedPlace( Pp ), Contains( cts:Pp, ctd:Pc ).
  (iii) matchingOffer( O, Sat ) :- O:TouristicOffer( destination:Dest, period:Per ),
    requestedPlace( Dest ), query( customer:Cust, period:Per ),
    preferenceSatisfaction( Cust, O, Sat ).

  % generate alert messages if matching offers are not recommendable
  (iv) badOffer( O ) :- matchingOffer( O ), O:TouristicOffer( period:Per, destination:D ),
    BadPeriod( place:D, period:Per ).
  (v) badOffer( O ) :- matchingOffer( O ), O:TouristicOffer( destination:D ),
    DangerousPlace( place:D ).

  % select alternative/recommended offers
  (vi) recommendedAlternative( Off, Sat ) :- query( customer:Cust, period:Per ),
    SuggestedPeriod( place:Dest, period:Per ),
    not DangerousPlace( place:Dest ),
    Off:TouristicOffer( destination:Dest, period:Per ),
    preferenceSatisfaction( Cust, Off, Sat ).
}

Figure 6. Personalized Trip Search: Welcome Suggestions and Tourism Offer Search.

The interested reader can find the details of the algorithms employed for evaluating logic programs and $\mathcal{L}_X$ descriptors in [32, 33] and [25, 35, 34], respectively.

4. System Architecture

The architecture of the iTravel system, which is depicted in Figure 1 (b), is made up of four layers: Data Layer, Information Layer, Knowledge Layer, and Service Layer. In the Data Layer the input sources are dealt with. In particular, the system is able to store and handle the most common kind of sources: e-mails,
plain text, pdf, gif, jpeg, and html files. The Information Layer provides ETL (Extraction, Transformation and Loading) functionalities, in particular: in the loading step the documents to be processed are stored in an auxiliary database (that also manages the information about the state of the extraction activities); whereas, in the Transformation step, semi-structured or non-structured documents are manipulated. First the document format is normalized; then, the “bi-dimensional logical representation” is generated (basically, the HiLX portions are identified); finally, HiLX descriptors are applied in the Semantic Extraction step and ontology instances are recognized within processed documents. The outcome of this process is a set of concept instances, which are recognized by matching semantic patterns, and stored in the core knowledge base of the system where the tourism ontology resides (Knowledge Layer). Domain ontology and extracted information are handled by exploiting the Persistency manager of the OntoDLV system (see Section 2.1). The Services Layer features the profiling service and the intelligent search (see Section 3.3) which implements the reasoning on the core ontology by evaluating in the OntoDLV system a set of logic programs. The Graphical User Interface (GUI) can access the system features by interacting with a set of web-services. The GUI offers both a dedicated access to travel agents and a user friendly environment for customers.

5. Experiments

In order to provide a concrete idea on the system behavior, we report, in this Section, the results of some experiments assessing “on-the-field” system performance in a concrete usage scenario.

Benchmark Data and Settings. We considered a corpus of 755 tourism leaflets received by travel agency TopClass, containing 10285 packages in total. Most of the extracted information was contained in the email attachments and stored in different formats (mainly pdf and html). The emails were sent by the following tour operators: Francorosso, Viaggidea, Kambola, Villaggi Bravo, Non solo weekend, Volacon, Cabo Verde Time, Brazil Time, Gastaldi, Itermar, Grimaldi, and Club Med. The system was run on a computer featuring an Intel T2500 CPU clocked at 2 GHz with 2GB of RAM, and a SATA 7.200rpm 8MB cache 1.5Gb/s HD. The system employs OntoDLV Version 1.6 configured with the latest release of DLVDB [40], and the DBMS MySQL 5.1. It is worth remarking that the experimental data are real-world tourism leaflets received by the travel agency, and also the hardware used for running the experiments is the one currently employed by TopClass for hosting the system. Thus, we report on the system performance on a real use case. Experiments were carried out by the project partner Exeura s.r.l.

Information Extraction. In order to obtain a perfect extraction against which comparing the output of the system, the corpus was first manually-inspected for building the results of an ideal extraction; that is, an extraction which is both sound and complete, where all relevant information are extracted from the leaflets in the right way. Then, the system was run on the same corpus and the result of the automatic extraction compared to the ideal one. The effectiveness of extraction is measured in terms of the classical notions of Precision (Pr), Recall (Re) and F-measure (Fa), which are defined as follows [36]:

\[
P_r = \frac{|TP_{a}|}{|TP_{a}| + |FP_{a}|}; \quad R_e = \frac{|TP_{a}|}{|TP_{a}| + |FN_{a}|}; \quad F_{a} = \frac{Pr \cdot Re}{(1-\alpha)Pr + \alpha Re}.\]

Here \(|TP_{a}|\) is the number of true positive extractions of an attribute \(a\) (for instance, for \(a=\text{destination}\) \(|TP_{a}|\) is the number of offers for which the destination has been correctly extracted), \(|FP_{a}|\) the number of false positive extractions of \(a\) (i.e., the
number of offers for which a wrong value for \( a \) has been extracted) and \( FN_a \) the number of false negative extractions of \( a \) (i.e., the number of offers in the ideal extraction where the automatic extraction missed the extraction of a value for \( a \)). Further, the parameter \( \alpha \in [0 \ldots 1] \) of the \( F^\alpha \)-measure was set to 0.5 to give the same importance to \( Pr \) and \( Re \). Figure 7(a) reports the extractor performance regarding the following attributes: destination, tour operator, cost. The crucial attribute destination and the tour operator are extracted very well in most cases (with an \( F^\alpha \)-measure of 0.84 and 0.85 respectively). The cost attribute is the most difficult to be extracted, since the same leaflet might contain several different costs depending on different combination of travel and accommodation options. Since the destination is, by far, the most frequently used attribute for search, the extraction module of iTravel revealed to be effective in practice and the travel agent using the system was fully satisfied by iTravel behavior.

**Time Performance.** The entire extraction process applied to the benchmark data required 10.1 seconds only, corresponding to the extraction of about 1000 offers per second. Since extraction is an off-line process, the time performance is fully satisfactory for the user. Concerning query performance, we have run three kind of search queries for offer retrieval based on destination and budget (destination only, budget only and destination + budget). Figure 7(b) reports the average execution times elapsed for answering package search queries. Results are averaged over 40 queries per kind. Note that the system required less than two seconds on the average in all cases, thus performing in a satisfactory way on real-world data.

### 6. Related Work

The usage of ontologies for developing e-tourism applications is already studied in the literature [24, 6, 7, 26, 27, 31], and the potential of the application of semantic technology recognized [12, 21, 4].

The architecture of an e-tourism system able to create a tourism package in a dynamic way is presented in [6]. This system permits the customer to specify a set of preferences for a vacation and dynamically access and query a set of information sources to find component such as accommodation, car rental, and leisure activities in real time. It is based on an ontology written in OWL-DL. The ontology used in [7, 6] encodes the same key concepts as ours, but does not include information about user preferences. The SPETA system [2], which is based on the ontology of [6], acts as an advisor for tourists. Fundamentally, SPETA follows people who need advising when visiting a new place, and who consequently do not know what is interesting to visit. Here the ontology is enriched with user profile information for determining the common characteristics of the previously visited places and the user behavior. In this way the system recommends attractions which are likely to fit the user expectations. It exploits GPS technology to know user position and it gets user data from previous users history and also from social networks. Both SPETA and iTravel use an ontology for building personalized solutions for
the users, but the goal of SPETA is different from that of iTravel. Indeed, the former is conceived for offering assistance and information to the user when they already are on a place, while the goal of iTravel is to assist the users in the selection of a holiday. Another advantage of the our approach is the possibility of developing ASP programs that reason on the data contained in the ontology. The DERI [10] E-Tourism Working Group at STI Innsbruck studies e-tourism solutions based on the Semantic Web technology. Their goal is to develop an advanced eTourism Semantic Web portal which will connect the customers and virtual travel agents from anywhere at any time with any needs and requests. In [37], they present OnTour an information retrieval system that exploits an RDF engine for storing the data regarding accommodation facilities of different types; whereas in [11] they present a content management system called OnTourism. The main goal of OnTourism is to implement a semantic search functionality on a document repository, where documents are created by call centre agents. Extraction of tourism information from documents is performed by using the Lixto Software [23]. Documents can be tagged (i.e., keywords are assigned to documents) or classified w.r.t. an ontology by the users. Statistical methods are also used to build relations between tags and ontology elements. The search functionality offered by the system combines the results of a semantic search (ontology based) with a search on the folksonomy tags.

AuSTO [20] system is a e-tourism planner presented at Enter conference in 2007. It provides facilities for planning trips using semantic web technologies and ontologies. It allows users to select travel preferences and develop a itinerary searching through offers inserted by several vendors. One of the main features of iTravel, i.e., automatic population of the offer class is not present in AuSTO, since in this system the information is directly inserted by vendors. Besides specific systems a number of ontologies has been proposed to deal with different challenges of e-tourism. AIFB [1] Ontology, developed for GET-ESS [39] project, models tourism domain. It is employed in a system performing text mining, storage and natural language querying; Travel information is crawled from web sources and semantically interpreted by GETESS system. The Harmonise Ontology [18], initially developed under an homonymous EU project IST-2000-29329, is now the core of HarmoNET (Harmonisation Network for the Exchange of Travel and Tourism Information), which is an international network for tourism data exchange. Harmonise is developed in RDF and is based on two main concepts: events and accommodation. The main goal of Harmonise is Semantic Interoperability (i.e., the seamless cooperation of heterogeneous information systems). Each member of the HarmoNET network provides a mapping mapping between their data model and the Harmonise one that acts like a mediator. In this way, network members can share its own information without changing local information systems. Both the structure and the final goal of the Harmonise ontology are different from the iTravel one, which is specifically conceived for travel agencies and tailored for searching pre-packaged tourism offers. Many other tourism ontologies are specified in OWL language [38] or in its ancestor DAML [8]. Among them, a relevant tourism ontology is QALL-ME [30], that models (in the OWL-DL language) some aspects of tourism domain (destinations, transportation means, etc.) and provides a shared infrastructure for multimodal and multilingual question answering. Hi-Touch is an OWL ontology realized by Mondeca [9], which is conceived for tourism object classification. Hi-Touch collects classical travel entities (destination, location, museum, etc.) and terms from the WTO [42] thesaurus. TAGA is a core ontology of a travel industry simulation game. It connects typical travel concepts with tourism events and activities. The EON [41] travelling ontology is a travel-oriented model developed in DAML. It is divided in two main concepts: static artifacts, containing non time dependent objects, and dynamic artifacts, related with time changing entities. It is worth noting

5For a more detailed comparison among available tourism ontologies we refer the reader to [31, 4].
that, as pointed out in [4], the vocabulary of available tourism ontologies covers a limited set of concepts and often describes the domain from different perspectives due to the restricted application scope from which the ontologies have been elicited. Indeed, all the mentioned ontologies share some general concepts (often modeled in a different way); whereas, some of them, deepen the description of some specific aspect of the tourism domain (depending on the specific needs of the application/project that inspired its realization). Clearly, none of them provides a solution for all the e-tourism systems challenges. The same considerations apply for the tourism ontology presented in this paper, which has been developed with the intent of allowing for personalized tourism-package search and automatic extraction of offers. The distinctive features of our tourism ontology (which are not present contemporarily in all the existing proposals) are: (i) the representation of customer preferences; (ii) the precise representation of the travel agent knowledge (that allows for driving the search); (iii) the suitability for automatic population from tourism leaflets; and, last but not least, (iv) the possibility of reasoning on top of the ontology by means of logic rules (and, in particular ASP programs). Note that, especially the fourth feature discouraged the adoption of OWL (and even OWL-DL) ontologies, since this language would not allow to fully exploit neither rule-reasoners nor the more stable and commercially developed relational databases. Indeed, whereas is acknowledged that rule-based inference systems are needed by OWL applications [17, 19], there is no accepted decidable solution for combining in the same framework [3] logic rules and OWL ontologies without sacrificing language expressiveness (even the simple unrestricted combinations of OWL-DL ontologies with Horn rules would lead to undecidability). This is not the case of our ontology which is specified in OntoDLP. OntoDLP is conceived for combining a rich ontology specification language with a powerful rule-based language (Answer Set Programming) which is suitable for developing enterprise-ontologies [33]. Similar considerations hold for the exploitation of database systems, indeed, OntoDLV transparently allows for exploiting the stable and widely adopted DBMS technology for storing and querying ontology data [5]; whereas the practical unsuitability of OWL-DL in this case is confirmed by the recent introduction of new ”profiles” that allows for database exploitation in the recent OWL 2 proposal [38]. However, those new OWL 2 profiles are obtained by both disallowing a number of constructors and limiting the constructors interactions, resulting in a compromised ontology language expressiveness. Note, however, that none of the above-mentioned OWL-based tourism ontologies complies with the database-oriented OWL 2 profiles. The reader interested in a complete comparison of OntoDLP and OWL can find more details in [32, 33]. Note also that, OntoDLV combines in a unified framework text-extraction and rule-based reasoning: the two key technologies dealing with the main challenges of the presented system.

7. Conclusion and Market Perspective

In this paper we have described the e-tourism system iTavel, a successful example of commercial and practical use of logic programming. The core of iTavel are (i) an ontology modeling the domain of the tourism offers, which is automatically populated by extracting the information contained in the e-mails sent by tour operators, and (ii), an intelligent search tool based on answer set programming is able to search the holiday packages that best fits the customer needs. Actually, several logic programs have been devised for implementing the intelligent search. In the development process, we exploit many of the advanced features of the language like negation as failure and aggregates. The declarative nature of ASP allows us to design effective solutions and to tune rapidly our modules by following the suggestions
of the domain experts. The system has been developed under PIA (Pacchetti Integrati di Agevolazione industria, artigianato e servizi) project “iTravel: Intelligent Tourism Advisor” funded by the Calabrian Region. The project team involved four organizations: the Department of Mathematics of the University of Calabria (which has ASP as one of the principal research area), Exeura srl (a company working on knowledge management), Top Class srl (a travel agency), and ASPIdea (a software farm specialized in the development of web applications). Each members exploited their specific knowledge for developing the key features of the system. The strong synergy among partners made it possible to push the domain knowledge of the travel agency TopClass in both the ontology and the reasoning modules. The result is a system that uses the knowledge of a seller of the agency and it is able to search in a huge database of automatically classified offers. iTravel combines the speed of computers with the knowledge of a travel agent for improving the efficiency of the selling process.

The iTravel system was initially conceived for solving the specific problems of a travel agency, and it is currently employed by one of the project partners: Top Class srl. We received very positive feedbacks from the market, indeed many travel agents are willing to use the system, and the potential of iTravel has been recognized also by the chair of the Italian Touring Club, which is the most important Italian association of tour operators. As far as future work is concerned, we are working on an enterprise version of the system conceived for offering its services to several travel agencies; moreover, we are investigating the application of data-mining techniques for automatically updating the user profile according to the customers’ buying history. The actual system is targeted for finding closed packages (as they are presented in the leaflets), and it does not features an automatic holiday composition from more packages (clearly, this task can be carried out manually by both the travel agent and the final customer). An automatic package-composition feature is also the subject of future work.

Acknowledgments. The authors are thankful to the staff of Top Class srl and ASPIdea srl; a special thanks is addressed to G. Candreva, G. De Franco, D. De Santo and C. Donato. This work was partially supported by the Regione Calabria and EU under POR Calabria FESR 2007-2013 within the PIA project of TopClass s.r.l.

References

[6] Cardoso, J.: Combining the semantic web with dynamic packaging systems, AIKED'06: Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases,


[23] Lixto: http://www.lixto.at/.


