Chapter 4
Emerging Semantic-Based Applications

Carlos Bobed, Roberto Yus, Fernando Bobillo, Sergio Ilarri, Jorge Bernad, Eduardo Mena, Raquel Trillo-Lado and Ángel Luis Garrido

In the last decade, we have witnessed the birth and spread of the so-called Semantic Web. From its initial proposal by Tim Berners-Lee (Berners-Lee et al. 2001; Shadbolt et al. 2006), to the latest trends and initiatives, such as Linked Data (Bizer et al. 2009a) and DBpedia (Bizer et al. 2009b; Mendes et al. 2012), the Semantic Web is progressively changing the landscape of the World Wide Web (WWW) through the use and adoption of the different semantic technologies that have come along with it. We can see how, although some of the goals of the Semantic Web have not been reached yet, several well-known and successful applications are already using semantic technologies, such as Google's Knowledge Graph, Microsoft's Satori, or Facebook's Graph Search.

In this successful scenario, ontologies have played a crucial role. Defined by Tom Gruber as "an explicit specification of a conceptualization" (Gruber 1993, 1995), ontologies allow to model and capture the semantics of different knowledge domains, providing a means to share definitions, and reach an implicit agreement on the meaning of the published information. Ontologies represent the vocabulary of some domain from a common perspective using a formal language, such as the current standard Web Ontology Language (OWL 2; Hitzler et al. 2012).

Being knowledge representation frameworks as they are, they might have an impact on many other different kinds of systems. In fact, they have already had; for example, ontologies have been successfully used in the integration of information and information systems (Mena and Illarramendi 2001; Wache et al. 2001). Thus, with their advance and the development of their associated technologies, we can

C. Bobed (✉) · R. Yus · F. Bobillo · S. Ilarri · J. Bernad · E. Mena · R. Trillo-Lado · Á. L. Garrido
Department of Computer Science & Systems Engineering, University of Zaragoza, C. María de Luna 1, 50018 Zaragoza, Spain

C. Bobed · R. Yus · F. Bobillo · S. Ilarri · J. Bernad · E. Mena · R. Trillo-Lado · Á. L. Garrido
e-mail: cbobed@unizar.es; ryus@unizar.es; fbobillo@unizar.es; silarri@unizar.es; jbernad@unizar.es; emena@unizar.es; raqueltl@unizar.es; garrido@unizar.es

M. Workman (ed.), *Semantic Web,* DOI 10.1007/978-3-319-16658-2_4

now explore deeper in the quest for smarter information systems which exploit the semantics of data.

Possible sources of enhancement for our applications could be the use of ontologies which are already shared and published, the possibility of modeling domains in detail, thanks to the expressivity of the different families of languages (mainly based on Description Logics (DLs; Baader et al. 2003)), or the exploitation of the huge amount of semantic data which have been already generated. In this chapter, we present different semantic-based applications and projects that we have developed in the Distributed Information Systems research group (SID, http://sid.cps.unizar.es) that currently benefit from these semantic technologies, and provide a good example of how the addition of semantics broadens the capabilities of an information system. In particular:

- In our approach to what could be considered as the most classical conception of the Semantic Web, we have applied the use of semantics to the field of keyword-based search. Using disambiguation techniques which exploit the knowledge stored in ontologies (Gracia and Mena 2008; Trillo et al. 2007a), we have developed two different approaches to keyword search over different information systems: *QueryGen* (Bobed 2013) and *Doctopush* (Trillo et al. 2011). The former one is oriented to perform semantic keyword-based search over heterogeneous information systems, proposing a generalized semantic keyword interpretation process, while the latter aims at performing semantic data retrieval over the WWW using the semantics of keywords to cluster relevant sources of information.
- We have also devised a framework to enhance semantically different tasks regarding information extraction (IE), such as automatic text classification, semantic search, and summarization of text sources, among others. This framework, called *GENIE* (GENeric Information Extraction Framework), aims at supplying a set of libraries designed to assist developers in projects of this nature, adopting a semantic approach to all modules, thus taking advantage of the latest advances made in ontological engineering and semantics.
- Regarding the standard formalism used for ontologies, we have studied an extension of the semantics of DLs to embrace Fuzzy Logic. This has led to the implementation of the fuzzy ontology editor *Fuzzy OWL 2* (Bobillo and Straccia 2011), and the fuzzy DL reasoners *DeLorean* (Bobillo et al. 2012a) and *fuzzyDL* (Bobillo and Straccia 2008), combining the expressivity of classical DL languages (which use crisp logic) with the flexibility of fuzzy logics for imprecise knowledge representation.
- Finally, we have studied the relationship between locations and semantics. On the one hand, we have applied the know-how acquired using ontological formalisms to study how we can model locations using different granularities while keeping and exploiting their semantics. On the other hand, we are currently working on enhancing the use of Location-Based Services (LBSs) adding semantics, which is crystallizing in *SHERLOCK* (Yus et al. 2014a), a system that searches and shares up-to-date knowledge from nearby devices to provide users with interesting LBSs.

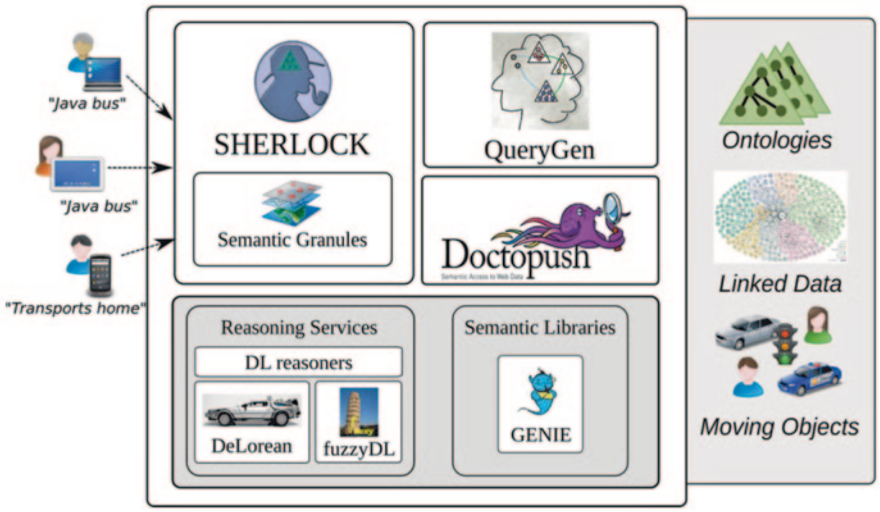


Fig. 4.1 Overview of our semantic-based systems

In fact, all of these systems are not isolated, but help each other to perform different tasks. In Fig. 4.1, we can see how they could be coordinated. SHERLOCK uses our different semantic models of locations to give meaning to the locations of the users, and infer further information out from them. Also, SHERLOCK might provide a keyword interface and could use QueryGen to construct a formal specification of the requested service out from the user's input keywords. Doctopush could be integrated as well, and used as a particular service under the umbrella of SHERLOCK.

All of these systems use several reasoning and semantic services, which could be abstracted in a separated layer. On the one hand, this layer would expose the services provided by classical DL reasoners, as well as the reasoning services of our fuzzy extension for DLs represented in Fig. 4.1. On the other hand, we could find the services exposed by GENIE to enhance different semantic tasks that they have to perform.

For the sake of readability, we devote the next section to give a brief overview of the basics on DL-based ontologies and DL (Baader et al. 2003) reasoners, which are thoroughly used in the presented systems. Then, in the rest of this chapter, we elaborate on these four points, all of which share the use of semantics (at different levels) as their main value.

4.1 Background

Ontologies (Frank 2003; Haarslev et al. 1998) are one of the most popular approaches to represent the knowledge of a domain. Although they can be developed using different languages and formalisms (Gómez-Pérez et al. 2004), we focus on the basics of OWL, the language that is the current W3C standard for representing ontologies in the Semantic Web, which has Description Logics as the underlying formalism. Thus, in DL-based ontologies, the basic ontological representation primitives (also called ontology elements) are individuals (or instances), concepts (or classes), properties (also called roles or relations), datatypes (or concrete domains), and axioms:

- *Individuals* are objects of the world. For instance, *John*.
- *Concepts* are sets of individuals. For example, *Human*. We denote the set of all concept names of an ontology by N_C.
- *Properties* define interactions between pairs of individuals of the domain, or between an individual and a datatype value. For example, *isParentOf* can be used to define that a human is the parent of another one, while *hasAge* can relate a human with her age.
- *Datatypes* represent concrete data values such as numbers (real, rational, integer, nonnegative, etc.), strings, booleans, dates, times, or XML literals, among many other possibilities.
- *Axioms* are formal conditions to be verified by the elements. An ontology can be seen as a finite set of axioms, usually divided in three parts: an assertional box (*ABox*), a terminological box (*TBox*), and a role box (*RBox*), with axioms about individuals, concepts, and roles, respectively. For example, an ABox can assert that *John* is a member of the concept *Human*, and a TBox can assert that *Man* is a subclass of *Human,* usually denoted $Man \sqsubseteq Human$.

Apart from the explicitly represented knowledge, it is possible to perform several reasoning tasks to deduce implicit knowledge, that is, logical consequences of the knowledge in an ontology. This is possible because OWL, as mentioned before, has a formal semantics based on DLs (Baader et al. 2003).

DLs are a family of logics for representing structured knowledge. They are a well-known formalism providing a good trade-off between expressivity of the representation and efficiency of the reasoning. Each DL is denoted by using a string of capital letters which identify its expressivity. For instance, the standard language for ontology representation OWL 2 is equivalent to the DL $\mathcal{SROIQ}(\mathbf{D})$. The expressivity of a DL translates into what kind of constructors can be used to form new concepts. For example, if letter C is in the expressivity of a DL, it means that it can use the constructor $\neg$ to express the contrary of a concept ($Woman \sqsubseteq \neg Man$, a woman cannot be a man). We summarize informally in Table 4.1 the most common constructors that lead to a DL with expressivity $\mathcal{ALC}$, while other logics and their allowed constructors are presented in Table 4.2. For more formal details, see Baader et al. 2003.

The most typical reasoning services in ontologies are designed to check:

- *Consistency:* An ontology O is consistent iff it has a model, that is, there is an interpretation satisfying every axiom in O.
- *Entailment:* O entails an axiom τ iff every model of O satisfies τ.
- *Concept satisfiability:* A concept C is satisfiable w.r.t. O iff it is not interpreted as the empty set in some model of O.

Table 4.1 Constructors and their meanings for $\mathcal{ALC}$ DL

Constructor	Meaning
$\top$	Any element
$\bot$	No element, empty set
A	Atomic concept
$\neg$C	Elements that are not in C
C $\sqcap$ D	Elements in C and D
C $\sqcup$ D	Elements in C or D
$\forall$R.C	Elements "a" such that if "a" is related with "b" by the property R, then "b" is in C
$\exists$R.C	Elements "a" that are related by property R with an element "b" in C

Table 4.2 Expressivity and complexity of reasoning in some important DLs

Logic	Expressivity	Complexity class
$\mathcal{AL}$	$\top, \bot, \neg A, \sqcap, \forall, \exists R.\top$	PTime
$\mathcal{ALC}(=\mathcal{ALUE})$	$\top, \bot, \sqcap, \sqcup, \forall, \neg, \exists$	ExpTime
$\mathcal{SHIF}$(**D**) (OWL Lite)	($\mathcal{S}$ =) $\mathcal{ALC}$ + transitive roles, role hierarchies ($\mathcal{H}$), inverse roles ($\mathcal{I}$), functional roles ($\mathcal{F}$), datatypes (**D**)	ExpTime
$\mathcal{SHOIN}$(**D**) (OWL DL)	$\mathcal{SHI}$, nominals ($\mathcal{O}$), nonqualified numerical restrictions ($\mathcal{N}$), datatypes (**D**)	NExpTime
$\mathcal{SROIQ}$(**D**) (OWL 2)	$\mathcal{SHOIQ}$(**D**), complex role inclusion ($\mathcal{R}$), self-restriction, and additional role axioms	N2ExpTime
$\mathcal{EL}^{++}$ (**D**) (OWL 2 EL)	$\top, \bot, \sqcap, \exists$, role hierarchies, nominals, concrete domains (use of constructors with syntactical restrictions)	PTime
DL-Lite (OWL 2 QL)	$\top, \bot, \sqcap, \exists, \neg$ (use of constructors with syntactical restrictions)	LOGSPACE
DLP (OWL 2 RL)	$\top, \bot, \sqcap, \sqcup, \forall, \neg, \exists$, cardinality cardinality restriction (0 .. 1) (use of constructors with syntactical restrictions)	PTime

- *Concept subsumption:* A concept D subsumes a concept C w.r.t. O iff C is interpreted as a subset of D in every model of O.
- *Classification:* The classification of an ontology O consists of computing a hierarchy of concepts based on their subsumption relation.

There are plenty of software applications (called reasoners) implementing ontology reasoning services. Some examples are JFact, HermiT (Glimm et al. 2014), or Pellet (Sirin et al. 2007).

4.2 Semantics Behind Keywords

The usage of keyword search has spread in the past few years thanks to its simplicity and its adoption by the main web search engines. Common users have found in it an easy way to express their information needs, defining their searches just by giving a plain set of keywords, and letting the system do all the work for them. However, the ease of use of keyword search comes from the simplicity of its query model, whose expressivity is low compared with other more complex query models (Kaufmann and Bernstein 2010).

Moreover, the use of keyword queries as starting point for information searches introduces a semantic gap between the user intention and the queries as, in fact, keyword queries are simplifications of the queries that really express the user's information need. Thus, there might be a gap between the posed query and the information that the users would like to obtain, for example, when talking about web searches, users usually have to browse the returned web pages looking for the needed information.

In this context, we also have to bear in mind that polysemous words introduce ambiguity in such queries, which cannot be solved without the intervention of the user. For example, if a user inputs the keyword "apple" in Google, locating information about the fruit with such a name will be difficult for her. (As of June 23, 2014, no hit about the fruit appears in the first 40 ranked positions provided as result, with the notorious exception of the page in Wikipedia, whose results are promoted.) Another different example of ambiguity could be the keywords "apache attack," where a user might be looking for information about the Apache helicopter or about how to secure an Apache server. One could argue that the ambiguity in these examples is due to the lack of input keywords, but experience tells us that the average number of keywords used in keyword-based search engines "is somewhere between 2 and 3" (Manning et al. 2008), which points out another problem of keyword search: Users tend to omit important keywords/information, as they consider them implicit in the query.

Being as useful as keyword-based searches have proved to be, we advocate for improving them by first establishing the proper meaning of each input keyword, which allows knowing exactly what the user is referring to with each of the keywords in the input set. This implies discovering possible meanings for each of the keywords, and disambiguating them to obtain their most probable one separately and as a whole set (i.e., the meaning of a keyword affects to the rest of keywords in the input set). In the rest of this section, we explain how we have applied this approach in two different systems which exploit the semantics behind keywords. First, we present QueryGen (Bobed 2013), a system that performs semantic keyword searches over heterogeneous information systems, interpreting keyword queries to access the underlying systems by taking into account the semantics of both the input keywords and the query languages involved. Then, we present Doctopush (Trillo et al. 2011), a pure Web-based search system, where the semantics of the keywords are used to categorize and group dynamically the results of a keyword query, providing more accurate and relevant information.

Semantic Keyword-Based Search: QueryGen

As we have seen before, using keywords as input language makes a system easier to use, but it implies that the queries that users can pose to the search system are limited by the lack of expressivity of this query model. Keyword queries are simplifications of the queries that really express the user's information need. Moreover, experience tells us that users tend to omit important keywords, as they consider them implicit in the query (recall that the average number of keywords used in keyword-based search engines "is somewhere between 2 and 3" (Manning et al. 2008)). However, the use of expressive formal languages such as SQL (ISO/IEC 2011) or SPARQL (Harris et al. 2013) is far from being easy for common users. What is more, the user must know the underlying schema and data she is accessing to effectively query it.

Thus, the sweet spot would be to mix the expressivity of formal languages with the ease of use of keyword queries, while making the user unaware of the data sources being accessed to solve her information needs. To reach this sweet spot, we advocate for a *semantic keyword-based search*, a keyword-based search process which takes into account the semantics of both keywords and query languages during the whole search process. Our objective is to discover and solve the user's information need taking as starting point a set of input keywords. We divide this task into three sub-objectives:

- To discover the exact meaning of each of the keywords in the set of input keywords.
- To give them an interpretation and express it into a formal language to capture the information need accurately.
- To access the proper information system/s transparently to the user, taking into account the different characteristics that the accessed systems might exhibit.

The first objective, the discovery of each keyword's meaning, allows us to work during the whole process with keywords with well-established semantics, which we call *semantic keywords*. The second one implies structuring a bag of keywords into a structured query, a process which is named *keyword query interpretation* (Fu and Anyanwu 2011). The achievement of the last objective is strongly helped by having the information formally expressed, allowing our system to access semantically even non-semantically enhanced data sources. The semantics behind the input set of keywords, the semantics of the different query languages, and the different semantics of the access models are considered to provide a flexible and efficient way to perform a semantic keyword search on heterogeneous information systems. Figure 4.2 shows the three main steps of the process, presenting further details on the disambiguation step.

Discovery of Keyword Senses First of all, we have to introduce the exact meaning of *sense* in our system: A sense is the precise meaning of a keyword in a context, that is, the meaning of a keyword is determined by its surrounding keywords. In our system, a sense is represented by a tuple formed by the term itself, an ontological context composed by a list of possible synonyms (with their URIs) and ontological

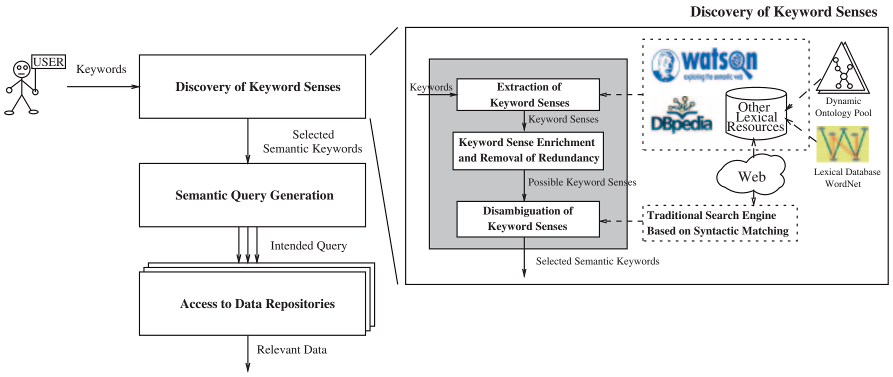

USER
Keywords
Discovery of Keyword Senses
Selected Semantic Keywords
Semantic Query Generation
Intended Query
Access to Data Repositories
Relevant Data
Discovery of Keyword Senses
Keywords
Extraction of Keyword Senses
Keyword Senses
Keyword Sense Enrichment and Removal of Redundancy
Possible Keyword Senses
Disambiguation of Keyword Senses
Selected Semantic Keywords
watson
DBpedia
Other Lexical Resources
Dynamic Ontology Pool
Lexical Database WordNet
Web
Traditional Search Engine Based on Syntactic Matching

Fig. 4.2 An overview of the whole process: From plain keywords to data access

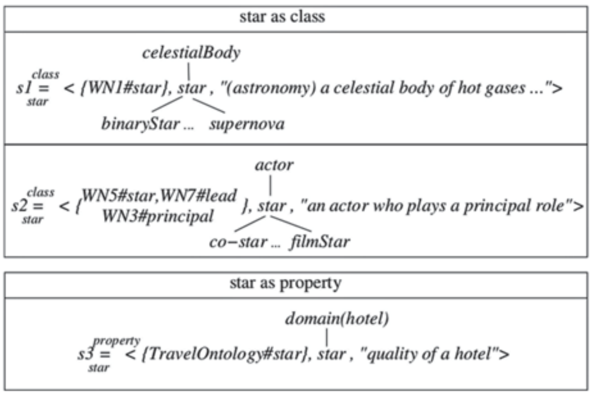

star as class
celestialBody
s1 class star = < {WN1#star}, star , "(astronomy) a celestial body of hot gases ...">
binaryStar ... supernova
actor
s2 class star = < {WN5#star, WN7#lead WN3#principal }, star , "an actor who plays a principal role">
co−star ... filmStar
star as property
domain(hotel)
s3 property star = < {TravelOntology#star}, star , "quality of a hotel">

Fig. 4.3 Possible senses for keyword *star*

information about the term, and a description in natural language. Each ontological context is built by integrating information from different ontologies. Figure 4.3 shows some possible senses for the user keyword *star*, retrieved from online ontologies.

So, our search starts by discovering and building these senses for the plain input keywords. Then, the discovery of the semantics behind each of the input keywords is done by taking into account their individual possible semantics as well as the possible semantics of their context (the rest of keywords), following the proposal in Trillo et al. (2007a). In particular, this process is divided into three substeps (see Fig. 4.2):

- *Extraction of Keyword Senses:* The system obtains the possible meanings of each keyword by consulting a dynamic pool of ontologies (in particular, it queries

Watson (d'Aquin et al. 2007), DBpedia (Mendes et al. 2012), WordNet (Miller 1995), and other ontology repositories to find ontological terms that syntactically match the keywords, or one of their synonyms). For each matching, the system builds a sense, which is semantically enriched with the ontological terms of the corresponding synonyms by also searching in the ontology pool. As a result of this step, we obtain a list of candidate keyword senses for each user keyword. In Fig. 4.3, three possible senses (two as a class and one as a property) retrieved for the user keyword *star* are shown.

- *Keyword Sense Enrichment and Removal of Redundancy:* In the sense list obtained in the previous step, there might be redundant meanings as the senses for each keyword are built with terms extracted from different ontologies. An incremental algorithm is used to align the different keyword senses and merge them when they are similar enough, and thus, to avoid redundancy. Our system calculates a *synonymy probability* that considers both linguistic and structural characteristics of the source ontologies: The linguistic similarity is calculated considering the different labels of each term as strings, and the structural similarity is calculated recursively by exploiting the semantics of the ontological context of the keyword sense until a certain depth. Finally, both similarity values are combined to obtain the resultant synonymy measure. The formulae for the synonymy for each type of senses (concepts, roles, and instances) can be found in Trillo et al. (2007a). Senses are merged when the estimated *synonymy probability* between them exceeds a certain threshold.[1] Thus, the result is a set of *different* possible senses for each user keyword entered.
- *Disambiguation of Keyword Senses:* A disambiguation process is carried out to select the most probable intended sense of each user keyword by considering the possible senses of the rest of keywords. The senses are compared by combining (Gracia and Mena 2009): (a) a Web-based relatedness measure that measures the co-occurrence of terms on the Web according to traditional search engines such as Google or Yahoo, (b) the overlap between the words that appear in the context, and the words that appear in the semantic definition of the sense (Banerjee and Pedersen 2003), and (c) the frequency of usage of senses (when available, as in WordNet annotated corpora). Our disambiguation process can be extended by including other different disambiguation algorithms such as the ones defined in Po (2009), as our approach does not depend on a specific disambiguation algorithm. This way, the best sense for each keyword will be selected according to its context. Note that this selection might require the user's feedback to select the most appropriate sense for each keyword in a semiautomatic way.

However, establishing the meaning of each keyword of the input is just the first step to obtain a proper interpretation, as several queries might be represented by a given set of keywords. For example, given the keywords *fish* and *person* meaning "a creature that lives and can breathe in water" and "a human being," respectively,

[1] In Gracia et al. (2009), the authors proposed several strategies to obtain this threshold and validated them via thorough experimentation.

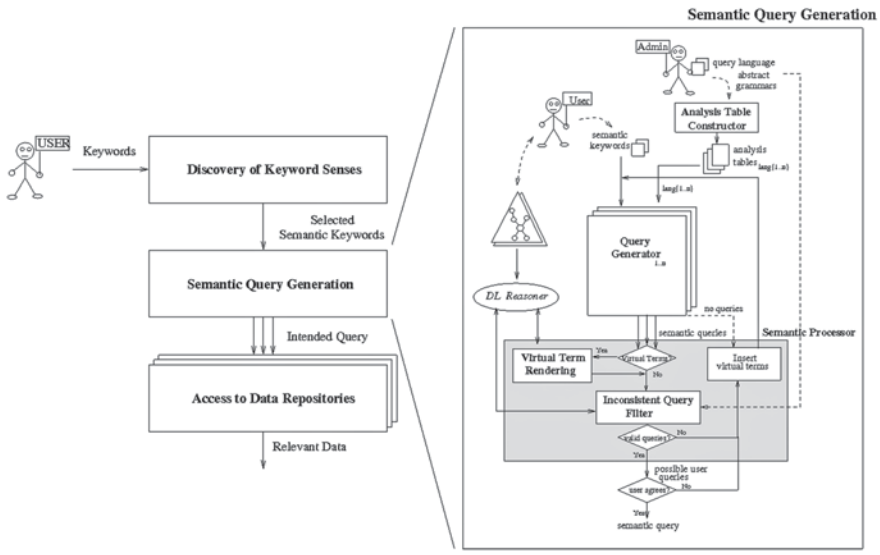


Fig. 4.4 Multi-language query generation process

the user might be asking for information about either biologists, fishermen, or even other possible interpretations based on those individual keyword meanings.

Semantic Query Generation The output of the previous step is a set of keywords which have their meaning properly attached, which we call *semantic keywords*. The ontological information that has been considered for obtaining the meaning of each keyword comes along with each of them. Our system automatically integrates this information, and then automatically builds a set of formal queries which, combining all the keywords, represents the possible semantics that could be intended by the user when she wrote the list of plain keywords. To do so, our system performs the following steps (see Fig. 4.4)

- *Analysis Table Constructor:* In order to capture formally the user's information need, the possible queries are expressed in the different query languages made available to our system, which are modeled using semantically annotated abstract grammars. These grammars lack syntax sugar and define: (1) how to combine the operators of a query language using *typed gaps*, that is, they specify which kind of queries can be built using concepts, roles, and instances in the corresponding query language (e.g., *And concept concept*), and (2) the semantics of the different operators giving their properties (e.g., associativity, symmetry, etc.) and DL expressions that will be checked with the help of a DL reasoner. The construction of the analysis tables (Aho et al. 2006) for the formal query languages is done off-line and just once for each language made available to our system.

- *Query Generator:* With the analysis tables, our system builds the possible queries for each query language according to its syntax. First, the Query Generator builds all the syntactically possible combinations according to the grammars of the available query languages. We call these combinations *abstract queries* because they have *gaps* that will be filled later with specific concepts, roles, or instances. These abstract queries are represented as trees, where the nodes are operators and the leaves are *typed gaps* (concept, role, or instance gaps). Then, for each abstract query tree generated, the gaps in the leaves are filled with the user keywords matching the corresponding gap type (i.e., keywords mapped to concepts are used to fill concept gaps in abstract queries, roles to fill role gaps, and so on). During this generation process, QueryGen takes into account the semantics of the different operators to avoid generating semantically equivalent queries.
- *Semantic Processor:* Then, once the set of syntactically possible queries is obtained, the Semantic Processor filters out the inconsistent ones with the help of a DL reasoner. This is done by using the DL expressions that specify each query language, which enables QueryGen to obtain an expression for the semantic consistency of each of the queries.

When no query satisfies the user, our system also performs a semantic enrichment of the input by adding *virtual terms*. They are generic typed gaps (to be replaced by concepts, roles, or instances) that represent the keywords that the user might have omitted, but without whom the intended query cannot be built. In a new query generation step, our system treats them as regular typed gaps but, instead of being replaced by input keywords, they are substituted by terms obtained from the ontologies which the input keywords were mapped to (during the previous discovery step). Thus, any query that the user could have in mind will be generated as a candidate interpretation as long as the available query languages used are expressive enough. Note how this query generation process has both a syntactic and semantic dimension: It generates only syntactically correct queries according to the grammar of each of the query languages made available to the system, and it takes into account the semantics of the operators of each language and the semantics of the keywords to avoid generating either duplicated or incoherent queries. This process is performed in parallel for each available query language, as their expressivity can differ from each other.

Access to Data Repositories Finally, once the user has validated the generated query that best fits her intended meaning, the system forwards it to the appropriate underlying structured data repositories (databases, Linked Data endpoints, etc.) that will retrieve data according to the semantics of such a query. This is not a trivial task, as our system must be able to adapt itself to their different query processing capabilities and access methods, and to their different data models and formats of the retrieved data. To provide QueryGen with enough flexibility to deal with this data heterogeneity, we advocate for the architecture shown in Fig. 4.5, whose main modules are the following ones:

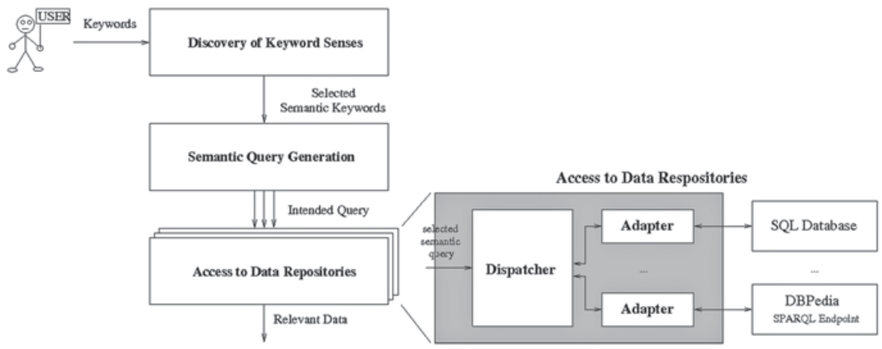

USER
Keywords
Discovery of Keyword Senses
Selected Semantic Keywords
Semantic Query Generation
Intended Query
Access to Data Repositories
Relevant Data
Access to Data Respositories
selected semantic query
Dispatcher
Adapter
Adapter
SQL Database
DBPedia SPARQL Endpoint

Fig. 4.5 Our system can retrieve data from different channels and data models

- *Dispatcher:* Once the user selects her intended query from those generated by QueryGen, the Dispatcher poses the query to the underlying data repositories that are capable of processing it. Every underlying system that is capable of processing the selected query is accessed in a parallel way as any of them could hold the desired answer. Finally, the Dispatcher correlates the data coming from the different systems and presents them to the user.
- *Adapter:* It wraps the access to the data stored in information systems with a certain data organization (e.g., there is an Adapter for relational databases, a different one for SPARQL endpoints, etc.). It registers itself in QueryGen providing information about the querying capabilities of the accessed information system, and making itself available to the Dispatcher. There is one instance of the appropriate kind of Adapter for each system accessed by QueryGen.

These *Adapters* are an evolution of the notion of *wrappers* used in OBSERVER (Mena and Illarramendi 2001), and encapsulate both the access methods and the actual syntax of the query languages and data formats, allowing QueryGen to abstract from them. Thus, we can add new information systems to feed QueryGen just by implementing and registering an appropriate Adapter in the system.

To sum up, QueryGen adopts an approach to the problem of keyword interpretation which provides a solution that, exploiting the semantics of all the elements that participate all along the search process, is flexible enough to deal with different data schemas (ontologies), different query languages, and different execution semantics. Moreover, using QueryGen, users can turn their information needs into formal queries without having to master the formal languages they are written in. Having formal queries instead of information needs removes the ambiguity and enables the systems to focus on answering the specific query that users would have posed if they knew how to write it. For further details on each of the different aspects, we refer the interested reader to Bobed (2013).

Semantic Data Retrieval: Doctopush

As the WWW evolved and became more and more popular, search services to locate web sites and pages have become indispensable for users. Broadly speaking, these search tools could be categorized (at first) as web directories-based ones, and web search engines. Web directories became less relevant than search engines because they do not scale properly due to the manual process required to classify the web pages and sites. So, the main research efforts were focused on web search engines, especially on those with keyword-based interfaces because of their ease of use and success. However, as mentioned above, the use of keyword queries to start off information searches introduces semantic gaps between the user information needs and the queries.

With the advent of the so-called Web 2.0, this need for efficient search tools increased, as users became content providers who often interact with other Web users, and thus the volume of content of the Web increased exponentially very quickly. To help users, keyword-based search engines specialized in different areas, such as job offers, books, etc., have been created in the last decade. They can be regarded as a hybridization of web directories and web search engines and are called *vertical search engines*. Some popular examples of them are Google Scholar (http://scholar.google.com, to search academic and research articles and papers), Technorati (http://technorati.com, to search blogs), and InfoJobs in Spain (http://www.infojobs.net, to search job offers). However, these search engines are developed ad hoc for each of the underlying domains, which can constitute a heavy barrier to the development of further ones.

Having into account the notion of web directories and vertical search engines, we propose to apply the disambiguation techniques previously described to group the hits retrieved by a traditional search engine into semantic categories. These categories are defined by the different meanings of the user's keywords and are used to categorize the retrieved links according to them, thus avoiding mixed-up results. Our approach, called Doctopush (Trillo et al. 2011), discovers the possible meanings of the keywords to create the categories dynamically in runtime by considering heterogeneous sources available on the Web. Differently from other proposals, such as clustering, the process of creation of categories is independent of the sources providing the results that must be shown to the users. Our approach considers two main steps: (1) discovering the semantics of the user keywords, and (2) semantics-guided data retrieval (see Fig. 4.6).

The first step, *Discovery of the Semantics of the User Keywords*, adopts the disambiguation technique previously described in QueryGen. The *Semantics-guided Data Retrieval* step pursues to provide the user with only the hits, retrieved by a traditional search engine, which she is interested in and filter out the irrelevant results. Thus, the system classifies the hits retrieved into the categories defined by the meanings of the user keywords. Moreover, the categories are ranked according to the interests of the user. In other words, the system attempts to select the hits that have the same semantics as the intended meaning of the user keywords and discard the others. This process is performed in four phases in runtime:

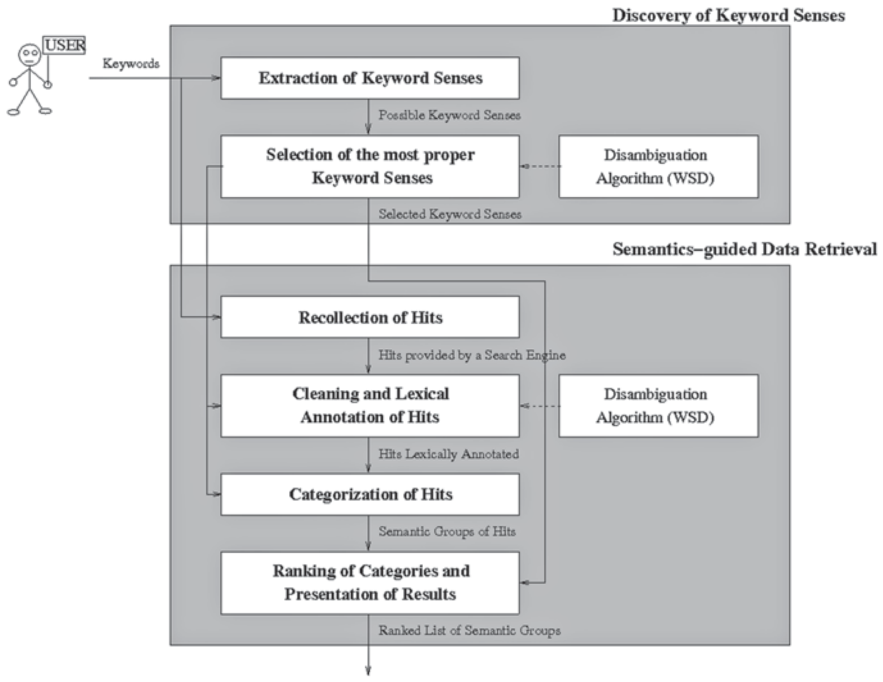

Discovery of Keyword Senses
USER
Keywords
Extraction of Keyword Senses
Possible Keyword Senses
Selection of the most proper Keyword Senses
Disambiguation Algorithm (WSD)
Selected Keyword Senses
Semantics–guided Data Retrieval
Recollection of Hits
Hits provided by a Search Engine
Cleaning and Lexical Annotation of Hits
Disambiguation Algorithm (WSD)
Hits Lexically Annotated
Categorization of Hits
Semantic Groups of Hits
Ranking of Categories and Presentation of Results
Ranked List of Semantic Groups

Fig. 4.6 Overview of Doctopush: A semantic prototype to group hits of a traditional search engine

- *Recollection of Hits:* The system performs a search in a traditional web search engine considering the user keywords as input. This search returns a set of relevant ranked hits, which represent the web pages where the keywords appear. The ranking of the hits depends on the specific techniques inner to the traditional search engine used (Google, Yahoo, Bing, etc.). Then, the hits returned are provided as input to the next phase (*Cleaning and Lexical Annotation of Hits*) incrementally, in *blocks of hits* of a certain size. In this way, new hits can be retrieved while the first blocks are being processed.
- *Cleaning and Lexical Annotation of Hits:* Each hit obtained in the previous phase (composed of a title, a URL, and a snippet) is automatically annotated lexically. Thus, first, each hit (H_j) goes through a *cleansing* process where stopwords are filtered out (creating the filtered hit H_j'). After that, the relevant words of the title and the snippet of each filtered hit H_j' are considered to perform its *lexical annotation*. A lexical annotation is a piece of information added to a term that refers to a semantic knowledge resource such as a dictionary, a thesaurus, or any other resource which represents a general or domain-specific ontology. So, for each filtered hit H_j', this process obtains a list of annotations denoted as $H_j' \rightarrow \{(S_1^x \rightarrow score_o, \ldots, S_1^y \rightarrow score_p), \ldots, (S_n^z \rightarrow score_q, \ldots)\}$. The list of annotations represents the senses which the user keywords are likely used within that hit.

Moreover, the score associated to each annotation indicates its reliability. This is needed because, in some situations, selecting only one sense for a user keyword in a certain hit is a difficult task even for a human due to the inherent ambiguity and polysemy. For example, a keyword can appear in the same hit with different senses (e.g., in the case of a hit corresponding to a dictionary entry), or two different senses for a keyword may have some overlapping (e.g., for the keyword *star*, the meanings are an "actor who plays a principal role" and "a performer who receives prominent billing"). Therefore, a list of annotations must be considered.

In our case, the annotation process is performed by considering each appearance of the user keywords in the filtered hit and its context (i.e., its relevant neighboring words), and by using the Web-based Word Sense Disambiguation (WSD) method and different configurations of the Probabilistic Word Sense Disambiguation (PWSD) method presented in Po (2009) and Po et al. (2009). For this, first, each appearance of a user keyword k_i in the title or the snippet of a filtered hit H_j' is marked with its probable senses, that is, $k_i^t \to \{S_i^x \to score_o, \ldots, S_i^y \to score_p\}$, where k_i^t denotes the tth appearance of k_i in H_j'. These annotations are used to perform the global annotation of the hit. Thus, when a user keyword sense S_i^x appears only once in the annotations performed, the sense and its corresponding score ($S_i^x \to score_o$) are incorporated to the list of annotations of the hit. Nevertheless, as a user keyword k_i could appear several times in H_j', the same user keyword sense S_i^x could appear in several annotations of k_i and have a different score in each of them; in this case, the maximum of these scores is considered for the global annotation of the hit.

- *Categorization of Hits:* The hits (already annotated as a result of the previous process) are grouped in categories by considering their lexical annotations. First, the system defines the categories that are going to be considered. Then, blocks of hits are classified. The potential categories are defined by considering all the possible combinations of candidate keyword senses of the input keywords, i.e., the Cartesian product of the candidate sense sets of the user keywords. For example, if the user introduces two keywords (k_1 and k_2) and, in the previous step, two senses are discovered for k_1 (S_1^1 and S_1^2) and one sense for k_2 (S_2^1), then the following potential categories are considered: $<S_1^1, S_2^1>$, $<S_1^2, S_2^1>$, $<U_1, S_2^1>$, $<S_1^1, U_2>$, $<S_1^2, U_2>$, and $<U_1, U_2>$, where U_1 and U_2 represent the unknown meanings considered for the keywords k_1 and k_2, respectively. Then, each hit is assigned to the categories defined by the meanings of the input keywords corresponding to the lexical annotation of that hit. So, depending on the scores of the meanings that are assigned to the user keywords in a hit, the hit could be classified in different categories at the same time, that is, the categories may overlap. Finally, the hits classified in a category are ranked according to their relevance for that category. That is, the system performs a *score-based ranking*. For this purpose, when a hit is assigned to a category, a score is also computed for the hit. This score is calculated by multiplying the scores associated to that hit for the different senses defining the category. Then, the hits within the category are ranked according to their scores (hits with the same score are ranked according to the order returned by the web search engine), as the hits in top positions are considered more relevant for that category.

- *Ranking of Categories and Presentation of Results:* Finally, the results of the Categorization of Hits phase are presented to the user. The system shows, in different tabs or category links, the categories considered that contain hits (Potential categories with no hits represent combinations of senses of the input keywords that are not detected in the hits collected). The order of the tabs or category links depends on the probability that the corresponding category represents the semantics that the user had in mind when she wrote her query. So, to rank the categories, three elements are considered: (1) the scores obtained previously, (2) the percentage of hits classified in the category, and (3) the position of the first hit in that category in the ranking provided by the web search engine. Thus, the global score for a category C_x is defined in the following way:

$$score(C_x) = \alpha score_{hitSenses} + \beta score_{\%hits} + \gamma score_{pos1stHit}$$

where α, β, and γ are the coefficients to tune the formula; $score_{hitSenses}$ is obtained by multiplying the scores (computed in previous phases) for the senses defining that category; $score_{\%hits}$ is equal to the number of hits assigned to the category C_x divided by the number of hits retrieved from the web search engine; and $score_{pos1stHit}$ is the inverse of the position of the first hit in C_x in the ranking provided by the traditional web search engine considered. Moreover, categories with unknown senses are considered less relevant, by assuming that the component $score_{hitSenses}$ is zero.

After developing the prototype, its performance under different contexts was evaluated in order to evaluate the interest of our proposal; for further details, see Trillo et al. (2011). We have also analyzed several related works in the following areas: clustering and categorization methods of documents, search engines that perform clustering of web documents, semantic search engines with the same goal as our proposal, and works about query reformulation and refinement. The main difference of our proposal with respect to other methods is that it considers the knowledge provided by ontologies available on the Web in order to dynamically define the possible categories for classifying the hits considered. Thus, it is independent of the sources providing the results that must be grouped.

4.3 Semantic Information Extraction: GENIE

The access to large amounts of information has become something regular in our daily life, and this has raised the need for more intelligent tools to collect, organize, analyze, and distribute all this information. These tools require capabilities that are not trivial and that can hardly be found in commercial products. To ease the development of such tools, it would be very useful to have off-the-self software that would comprise different elements to tackle these problems from different points of view under a common framework, providing solutions to many usual processes related to the extraction of information. This would help to increase productivity of organizations, and save resources needed to achieve their goals.

In this context, when it comes to handling information in nonstructured documents, many tasks are still open research problems. Among these tasks, we can find, for example, automatic text classification, summarization of text sources, extraction of data from raw text, or synthesis of knowledge out from natural language documents. The implementation of software to face these kinds of issues is not at all a trivial matter. Thus, in our research group, we have applied the know-how acquired developing different semantic information systems to develop a framework to help these tasks, called the GENIE project.

GENIE is the acronym for GENeric Information Extraction Framework. According to Russell and Norvig (Russell and Norvig 2003), Information Extraction (IE) means automatically retrieving certain type of information from natural language text. They say that IE is halfway between Information Retrieval (IR) systems and text understanding systems. GENIE is an architectural proposal that implements a set of components whose objective is to provide tools to make IE easier for the developers, integrating Semantic Web techniques with Machine Learning, Artificial Intelligence (AI) techniques, and Natural Language Processing (NLP) tools (Smeaton 1999). In particular, it supplies a set of libraries designed to assist developers in projects of this nature. An important feature that distinguishes this project from other similar works is the semantic approach given to all the modules, taking advantage of the latest advances made in ontological engineering and semantic technologies. To sum up, these are the main goals of GENIE:

- To create a framework able to handle different languages, and to integrate a large number of processes related to IE.
- To integrate into this framework, modular, generic, and open tools that can be used in other external applications.
- To develop an open framework allowing future expansion.
- To facilitate experimentation and testing allowing the improvement of current methods, and the development of new tools that represent an innovation in the field of IE.

The architecture of the GENIE framework is composed of a set of modules that implement essential tasks to execute different semantic IE processes. GENIE consists of a set of high-level units which can be orchestrated to form a semantic IE workflow by communicating with each other using XML. These units, if necessary, can be transformed into services, libraries, or web services, depending on the degree of decoupling and performance required. Specifically, GENIE is constituted by the following units:

- *Multilingual natural language analysis unit:* The aim of this unit is to provide basic information from raw texts. This information is similar to the information obtained by tools like morphological taggers or syntactic parsers.

- *Named Entities Detector:* This unit provides a specific semantic analyzer focused on Named Entities (NE; Sekine and Ranchod 2009), as they are subject of intense research in the context of IE. The recognition of NE has been enhanced with NLP and semantic methods which improve the term disambiguation.
- *Machine Learning unit:* This unit is another important piece inside the GENIE architecture, as it provides several unsupervised learning methods. Apart from Support Vector Machines (SVM; Joachims 1998), which have provided us very good results, this unit can also manage other unsupervised learning methods like clustering and statistical models like Bayesian networks. We have paid special attention to enhance the interconnectivity of these techniques with the rest of the units to provide a unified development framework.
- *Geographic information extractor:* Geographical resources have a special treatment due to their importance. Typically, when categorizing a text, about 30 % of the labels used by a documentation department are related to places. Thus, as detailed in Garrido et al. (2013b), we have incorporated the use of ontologies to enhance the geographical tagger service provided by this unit.
- *Categorization unit:* The goal of this unit is to automatically classify documents (Garrido et al. 2011, 2012). This unit uses most of the functions supplied by the aforementioned units.
- *Semantic analyzer:* This unit obtains information about the connotation of terms, that is, the real meaning of words in a context. Moreover, it searches relations of these words with others (e.g., synonyms, antonyms, hypernyms, hyponyms, etc.) using both NLP tools and ontologies to do this task. Finally, it is also used in order to disambiguate terms.
- *Automatic query expansion unit:* This unit provides an enhanced keyword-based search by expanding automatically the input query taking into account the semantic contents of the keywords and their relationships. The approach of this GENIE's unit consists of three steps: (1) *obtain words with common lemmas*, to extract those words that belong to the same family as the keywords entered as a query; (2) *obtain words with similar meanings*, to return records which contain words that are synonymous to the keywords introduced, using the previous units to analyze and to disambiguate the terms; and finally, (3) *refine the queries with NE*, by detecting and considering NEs as a whole. Further details are provided in Granados-Buey et al. (2014a, b).
- *Aggregation unit:* This unit provides personalized reports from raw text sources. These reports are elaborated from a set of predefined templates that define the presentation of the information of some types of results (e.g., people, companies, events, etc.). The sources can be documents, databases, triplestores, or even Linked Data. Some details of this unit can be found in Garrido et al. (2014a).

Our main contribution with GENIE is not only the effort of packing these different work units but also the use of semantic techniques for tasks which are usually tackled using purely linguistic approaches or machine learning. The advantages of joining machine learning approaches with semantic tools have been widely studied in Garrido et al. (2014c). Finally, another useful feature of GENIE is that it can incorporate new resources (databases, gazetteers, dictionaries, thesauri, RDF/OWL files, etc.) to enrich the service provided by work units. This incorporation may be progressive and is performed using standard formats for resources.

Regarding other existing projects with similar goals as GENIE's, we have to mention GATE (Cunningham et al. 2002). GATE is a Java suite of tools developed at the University of Sheffield, which began in 1995 and is used nowadays by a large community of scientists, companies, teachers, and students for NLP tasks. GATE is inspired by previous projects such as ATLAS (Bird et al. 2000). GATE is, on the one hand, an integrated development environment, and, on the other hand, a framework including a set of software building blocks ready to be used. It is a mature tool, very powerful and complete, working very well with English, German, and French. However, this framework does not give the same kind of support for languages such as Spanish, Italian, or Portuguese. In any case, the main difference between GENIE and GATE is the level where each project works at. GENIE is not only a set of tools ready to be used individually but also a collection of units prepared to be connected to other systems and resources, with a level of encapsulation and coupling adaptable to the needs of each project. In fact, GENIE could integrate a suite like GATE as another working unit.

Practical Applications. In addition In addition to its high interest as a research platform, this software has a lot of practical applications:

- *Search engines:* We have increased the performance of a standard index of a term-based search engine, making its behavior closer to a semantic search one. This allows to get results in spite of the fact that gender (male or female), number (singular or plural), or verb forms are different from the keywords used in the query. It also considers synonyms and related words when retrieving data, which greatly enhances the user experience. Further details can be found in Granados-Buey et al. (2014b).
- *Documentary collections:* GENIE can help to improve the productivity of, for example, administration staff or a documentation department, by automating tedious tasks of labeling. It has tools capable of categorizing documents using semantic tags with a high-precision level. Moreover, through a suitable interface, it could also update item details in the document database when necessary. A GENIE module has been linked to geographical databases, making it capable of taking into account tagged text locations and disambiguating it when needed. Finally, GENIE is also able to produce summaries of text with a defined length, something very useful in many areas. Hypatia (Garrido et al. 2014a) is an ongoing project that brings together all of these functionalities.
- *Information management:* With the ability of "understanding" a text, GENIE can extract information from a document and transform it to measurable values. This is very interesting as it can be applied to the analysis of reports, to obtain brand reputation, or to implement filters, among others. This extracted information can be captured and represented as a knowledge model. In particular, GENIE has been used to develop tools like (Garrido et al. 2013), a system able to generate

Topic Maps (Pepper and Moore 2001), a simple form of knowledge representation, out from plain texts.

- *Recommendation systems:* Today, it is quite common for e-commerce and bookmarking web sites to include some type of recommendation module that is able to identify and present items appealing to their users. Many techniques related to areas such as machine learning, IR, or NLP, among others, have been adopted to develop systems that recommend items like books, songs, or movies, for example. Even though recommendation systems have been developed for the past two decades, existing recommenders are still inadequate to achieve their objectives and must be enhanced to generate appealing personalized recommendations effectively. In this context, we have already proposed two approaches to recommender systems, SOLE-R (Garrido et al. 2014b) and TMR (Garrido and Ilarri 2014), which exploit the semantic services provided by GENIE.

The outcomes obtained by this system on real environments are very promising, and, in fact, this framework is already being used in real production environments, providing very good results; thanks to the use of semantic techniques.

4.4 Technologies for Fuzzy Knowledge

Despite the advantages of ontologies, it has been widely pointed out that they are not appropriate to deal with imprecise and vague information, which is inherent to several real-world domains. Let us discuss now some examples. The domain of medicine contains a lot of imprecise terms, and classical ontologies are not suitable to express that a patient is *slightly* unconscious or that anaphylaxis is *quite similar* to sepsis. Location-based applications are based on potentially imprecise sensor data: For instance, GPS sensors provide an approximate location. The domain of accommodations includes vague terms to categorize different lodging types (such as *Guesthouse*). For example, it is usually assumed that a *Guesthouse* is a "cheap, small, and more hospitable hotel", but the notions of *cheap, small*, and *hospitable* are clearly not well defined, having unsharp boundaries. Hence, the nature of the concepts in this domain makes crisp definitions unsuitable. For more applications of fuzzy ontologies, see Bobillo (2008).

Fuzzy set theory and fuzzy logic have proved to be suitable formalisms to handle this imprecise and vague knowledge (Zadeh 1965). In fuzzy logic, the usual convention prescribing that a statement is either true or false is changed. Every statement holds with a *degree of truth* measured on an ordered scale that is no longer $\{0, 1\}$ but (usually) $[0, 1]$. The main concept in fuzzy logic is that of fuzzy set. Essentially, the elements of a fuzzy set have degrees of membership valued in $[0, 1]$. For example, Fig. 4.7 shows a fuzzy set representing young people.

Fuzzy logics provide compositional calculi of degrees of truth. The conjunction, disjunction, complement, and implication operations are performed in the fuzzy case by a t-norm function $\otimes$, a t-conorm function $\oplus$, a negation function $\ominus$, and an implication function $\Rightarrow$, respectively. A quadruple composed by a t-norm, a t-

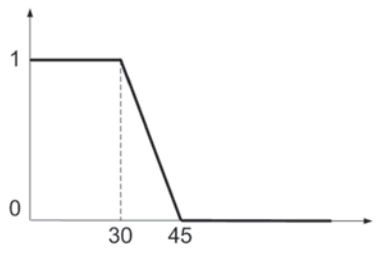


Fig. 4.7 Fuzzy set representing young people

conorm, an implication function, and a negation function determines a fuzzy logic. Table 4.3 shows the definition of some popular fuzzy logics, where $\alpha,\beta \in [0,1]$ are two degrees.

Fuzzy Ontologies Given the success and popularity of fuzzy logic, it should come as no surprise that several fuzzy extensions of ontologies can be found in the literature. The elements of a fuzzy ontology are extended as follows:

- *Fuzzy concepts* are interpreted as fuzzy sets of individuals, so the membership of an instance to a concept is a matter of degree. For example, *YoungHuman* can contain the fuzzy set of young people.
- Similarly, *fuzzy properties* between two individuals are interpreted as fuzzy relations, so pairs of elements are related to some degree. For instance, the property *isFriendOf* makes it possible to represent the degree of friendship between pairs of individuals.
- Now, it makes sense to consider *fuzzy axioms,* since statements are not either true or false but hold to some degree. For example, we can state that *john* belongs to the concept of *YoungHuman* with at least degree 0.9, meaning that he is rather young. Classical axioms τ are generalized as (τ,α).
- Finally, it makes sense to consider *fuzzy datatypes* generalizing crisp values by using a fuzzy membership function. For example, instead of considering the crisp value 18, now it is possible to consider *about18*. The former datatype is incompatible with the value 17.99, whereas the latter one is not.

Table 4.3 Truth combination functions of various fuzzy logics

Operator	Lukasiewicz logic	Gödel logic	Product logic	Zadeh logic
$\alpha \otimes \beta$	$\max(\alpha+\beta-1, 0)$	$\min(\alpha, \beta)$	$\alpha \cdot \beta$	$\min(\alpha, \beta)$
$\alpha \oplus \beta$	$\min(\alpha+\beta, 1)$	$\max(\alpha, \beta)$	$\alpha+\beta-\alpha \cdot \beta$	$\max(\alpha, \beta)$
$\alpha \Rightarrow \beta$	$\min(1-\alpha+\beta, 1)$	$\begin{cases} 1 \text{ if } \alpha \leq \beta \\ \beta \text{ otherwise} \end{cases}$	$\min(1, \beta/\alpha)$	$\max(1-\alpha, \beta)$
$\ominus \alpha$	$1-\alpha$	$\begin{cases} 1 \text{ if } \alpha = 0 \\ 0 \text{ otherwise} \end{cases}$	$\begin{cases} 1 \text{ if } \alpha = 0 \\ 0 \text{ otherwise} \end{cases}$	$1-\alpha$

The reasoning services in fuzzy ontologies include the same tasks for classical ontologies (see Section 4.1) together with some new ones. The most typical tasks are:

- *Consistency:* An ontology O is consistent iff it has a model, i.e., there is a fuzzy interpretation satisfying every axiom in O.
- *Entailment:* O entails an axiom τ iff every model of O satisfies τ.
- *Concept satisfiability:* A concept C is satisfiable to at least degree α (or α-satisfiable) w.r.t. O iff there is a model of O satisfying $(C(x),\alpha)$, for a new individual $x \notin O$.
- *Concept subsumption:* A concept D subsumes a concept C to at least degree α (or α-subsumes) w.r.t. O iff every model of O satisfies the axiom $(C \sqsubseteq D,\alpha)$.
- *Best entailment degree* (*BED*): The BED of a crisp axiom τ w.r.t. O is defined as the supremum of the degrees α such that O entails the axiom (τ,α), where $sup\ \emptyset = 0$.

Fuzzy ontologies require the development of new languages, reasoning algorithms, and tools. Unfortunately, they have not achieved yet the maturity of crisp ontologies, and additional research on this topic is still being carried out. However, there exist several approaches in the literature and implementations worth to mention. This section summarizes the main achievements developed by members of our research group in collaboration with international experts.

There is no standard language to represent fuzzy ontologies, and the different proposals have differences in the elements that are being fuzzified. For example, not all the approaches consider the definition of fuzzy sets using trapezoidal membership functions. To assist users in the process of fuzzy ontology building, there is a Protégé plug-in called *Fuzzy OWL 2* (http://webdiis.unizar.es/~fbobillo/fuzzyOWL2) that can be used to create and edit fuzzy ontologies (Bobillo and Straccia 2011). The plug-in uses a relatively abstract fuzzy ontology representation that can be exported into the particular syntax of different ontology languages.

Among the many existing implementations of fuzzy ontology reasoners, we will highlight two: *fuzzyDL* (http://webdiis.unizar.es/~fbobillo/fuzzyDL; Bobillo and Straccia 2008) and *DeLorean* (http://webdiis.unizar.es/~fbobillo/delorean; Bobillo et al. 2012). The former system implements a fuzzy extension of a classical tableaux algorithm. The latter follows an alternative approach and transforms a fuzzy ontology into an equivalent non-fuzzy ontology, preserving the semantics of the knowledge in such a way that it is possible to reuse the existing ontology reasoners. All the existing fuzzy ontology reasoners are complementary, because up to now, they support different fuzzy ontology elements, and hence cannot be easily ranked.

The rest of this section is dedicated to a deeper overview of these three applications: Fuzzy OWL 2, fuzzyDL, and DeLorean.

Modeling Fuzzy Ontologies with Fuzzy OWL 2

Fuzzy OWL 2 is a Protégé plug-in that makes it possible to develop fuzzy ontologies. Once the plug-in is installed, a new tab in Protégé, named Fuzzy OWL, enables to use it.

The plug-in is based on a methodology for fuzzy ontology representation using OWL 2 (Bobillo and Straccia 2011). The key idea of this representation is to use an OWL 2 ontology, and extend its elements with annotations representing the features of the fuzzy ontology that OWL 2 cannot directly encode. To separate the annotations including fuzzy information from other annotations, a new annotation property called *fuzzyLabel* is used, and every annotation is identified by the tag *fuzzyOwl2*. Since typing such annotations is a tedious and error-prone task, the plug-in makes the syntax of the annotations transparent to the users.

Figure 4.8 shows the available options of the plug-in: fuzzy queries, fuzzy ontologies, fuzzy modifiers, fuzzy concepts (fuzzy nominals, fuzzy-modified concepts,

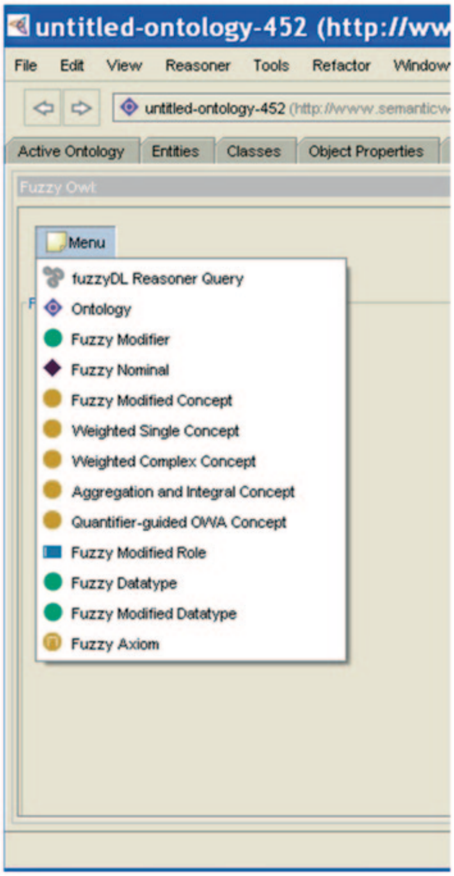
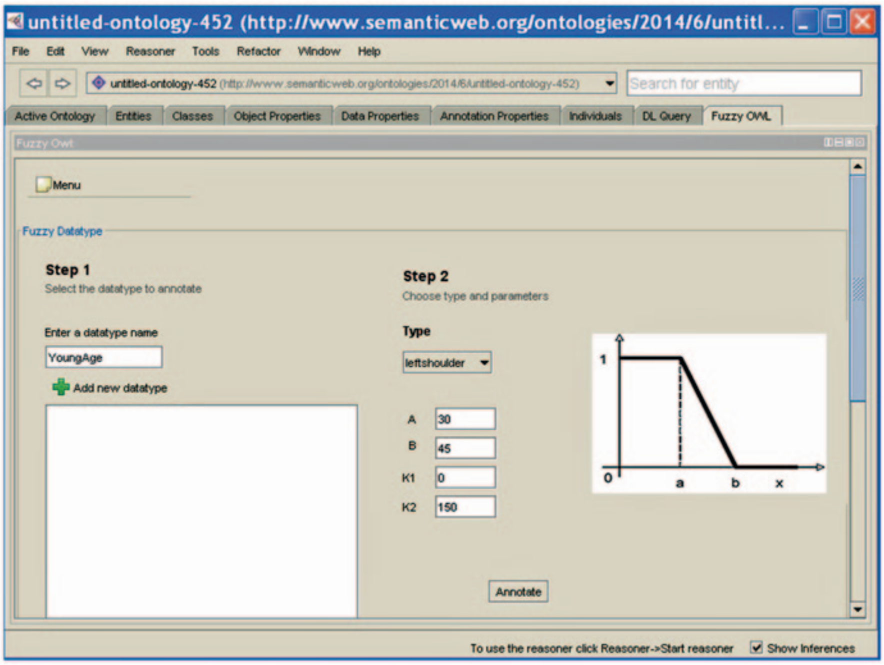


Fig. 4.8 Fuzzy OWL 2: Menu options


untitled-ontology-452 (http://www.semanticweb.org/ontologies/2014/6/untitl...
File Edit View Reasoner Tools Refactor Window Help
untitled-ontology-452 (http://www.semanticweb.org/ontologies/2014/6/untitled-ontology-452)
Search for entity
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals DL Query Fuzzy OWL
Fuzzy Owl
Menu
Fuzzy Datatype
Add new datatype
Annotate
1
0
a
b
x

Step 1
Select the datatype to annotate

Enter a datatype name: YoungAge

Step 2
Choose type and parameters

Type: leftshoulder

A: 30
B: 45
K1: 0
K2: 150

To use the reasoner click Reasoner->Start reasoner ✓ Show Inferences

Fig. 4.9 Fuzzy OWL 2: Creation of a fuzzy datatype

weighted single concepts, weighted complex concepts, aggregation and integral concepts, quantifier-guided OWA concepts), fuzzy properties (fuzzy-modified roles), fuzzy datatypes (including fuzzy-modified datatypes), and fuzzy axioms. A description of all the elements of a fuzzy ontology is out of the scope of this chapter; for more details, we refer the reader to Bobillo and Straccia (2011, 2013).

The non-fuzzy part of the ontology can be created by using Protégé as usual. After that, the user can define the fuzzy elements of the ontology by using the plug-in, namely fuzzy axioms, fuzzy datatypes, fuzzy modifiers, fuzzy-modified concepts, and fuzzy nominals.

Figure 4.9 illustrates the plug-in use by showing how to create a new fuzzy datatype. The user specifies the name of the datatype and the type of the membership function. Then, the plug-in asks for the necessary parameters according to the type. A picture is displayed to help the user recall the meaning of the parameters. After some error checks, the new datatype is created and can be used in the ontology.

Fuzzy ontologies developed with the plug-in use an independent syntax that does not align with any particular reasoner. Once the fuzzy ontology has been created, it has to be translated into the language supported by a specific fuzzy DL reasoner. For this purpose, the plug-in includes a general parser that can be customized to any reasoner by adapting a template code. The parser browses the contents of the ontology with *OWL API 3* (Horridge and Bechhofer 2011), which allows iterating over the elements of the ontology in a transparent way and prints an informative message. The template code has been adapted to build two parsers, one for fuzzyDL and one for DeLorean. Furthermore, similar parsers for other fuzzy DL reasoners can be easily obtained. To do so, one can just replace the default messages by well-formed axioms, according to the desired fuzzy ontology syntax.

Reasoning with Fuzzy Ontologies Using fuzzyDL

From a historical point of view, *fuzzyDL* can be considered as the first fuzzy DL reasoner. It is very popular and has been used in several applications. The supported language is thus a fuzzy extension of $\mathcal{SHIF}(\mathbf{D})$. It is also possible to use linguistic labels as degrees of truth, such as *high*, instead of forcing fuzzy ontology developers to use numbers in [0, 1]. This makes it possible to deal with unknown degrees of truth.

Apart from extending the elements of crisp ontologies to the fuzzy case, fuzzyDL introduces new elements specific from the fuzzy case, such as concept modifiers (using linear hedges and triangular functions), explicit definitions of fuzzy concepts (by means of triangular, trapezoidal, left-shoulder and right-shoulder functions), or some concept constructors (fuzzy rough concepts (Bobillo and Straccia 2012), aggregation operators, modified concepts, and threshold concepts). It is also possible to express linear inequations involving degrees of truth. The semantics is given by Zadeh and Lukasiewicz fuzzy logics, and some operators of Gödel fuzzy logic are also supported. Connectives of different fuzzy logics can be combined.

fuzzyDL supports standard reasoning tasks namely, consistency, concept satisfiability, maximum degree of satisfiability of a concept, entailment, concept subsumption, BED of an axiom, maximum degree of satisfiability of an axiom, and instance retrieval. Furthermore, it also supports some nonstandard tasks, such as variable optimization and different types of defuzzification of fuzzy sets.

The reasoning algorithm is based on a mixture of tableau rules and an optimization problem. All the reasoning tasks are reduced to the BED of a concept assertion. Then, it applies some satisfiability preserving tableau rules that not only decompose complex concept expressions into simpler ones but also generate a system of inequation constraints. These inequations have to hold in order to respect the semantics of the fuzzy DL constructors. After all the rules have been applied, an optimization problem must be solved before obtaining the final solution.

fuzzyDL implements several optimization techniques, such as general concept inclusion (GCI) absorption, concept simplification, lexical normalization, optimized use of n-ary conjunction and disjunction, advanced blocking techniques, normalization of degrees of truth, encoding of string names using integers, etc.

Although fuzzyDL can be freely downloaded, the user needs a *Gurobi* solver license because it is required to calculate the solutions of mixed integer linear programming (MILP) problems. fuzzyDL can be used as a stand-alone application, accessed by other applications by means of a Java API, or queried through the Fuzzy OWL 2 plug-in.

Reasoning with Fuzzy Ontologies Using DeLorean

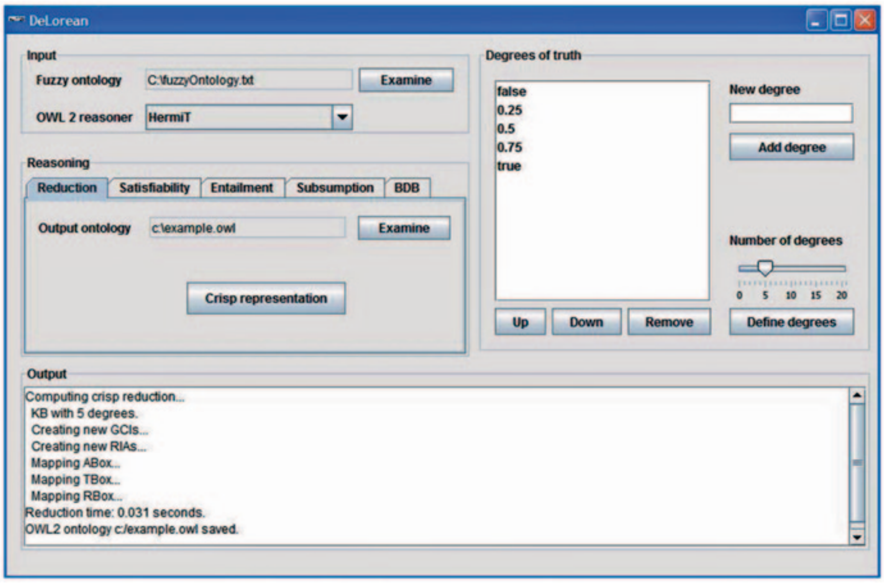
DeLorean (DEscription LOgic REasoner with vAgueNess) is a fuzzy ontology reasoner that supports fuzzy extensions of the languages OWL and OWL 2. Nowadays, DeLorean is the only reasoner that currently supports fuzzy OWL 2, although it does not support several elements supported by other fuzzy DL reasoners.

The reasoning algorithm is based on a reduction to reasoning in crisp ontologies detailed in Bobillo et al. (2012b). A consequence of the reduction is the possibility to reuse classical resources: editors, tools, reasoners, etc. In a strict sense, DeLorean is not a reasoner but a translator from a fuzzy rough ontology language into a classical ontology language (the standard language OWL or OWL 2, depending on the expressivity of the original ontology). Then, a classical DL reasoner is employed to reason with the resulting ontology. Nevertheless, due to this ability of combining the reduction procedure with the classical DL reasoning, we refer to DeLorean as a reasoner.

The supported language is thus a fuzzy rough extension of $\mathcal{SROIQ}(\mathbf{D})$. It is also possible to use linguistic labels as degrees of truth. The semantics is given by Zadeh and Gödel fuzzy logics, and connectives of both logics can be combined. The following reasoning tasks are supported: Computation of an equivalent crisp representation of the fuzzy ontology, consistency, concept satisfiability, maximum degree of satisfiability of a concept, entailment, concept subsumption, and BED of an axiom. There are some optimizations of the reduction to crisp $\mathcal{SROIQ}(\mathbf{D})$, but the only existing optimizations for reasoning in crisp $\mathcal{SROIQ}(\mathbf{D})$ are those implemented by the reused classical reasoner.

DeLorean can be used as a stand-alone application. In addition, DeLorean reasoning services can also be used from other programs by means of a Java API. In this section, we focus on the use of the reasoner through its graphical interface, which as illustrated in Fig. 4.10, is structured in four sections:

- *Input*. Here, the user can specify the input fuzzy ontology and the DL reasoner that will be used in the reasoning. The possible choices are HermiT (Motik et al. 2009), Pellet (Sirin et al. 2007), and any OWLlink-complaint reasoner. Once a fuzzy ontology is loaded, the reasoner will check that every degree of truth that appears in it belongs to the set specified in the section on the right.
- *Degrees of truth*. The user can specify here the set of degrees of truth that will be considered. 0 (false) and 1 (true) are mandatory. Other degrees can be added, ordered (by moving them up or down), and removed. For the user's convenience, it is possible to directly specify a number of degrees of truth, and they will be automatically generated.
- *Output*. Here, output messages are displayed. Some information about the reasoning is shown here, such as the time taken, or the result of the reasoning.
- *Reasoning*. This part is used to perform the different reasoning tasks that DeLorean supports. The panel is divided into five tabs, each of them dedicated to a specific reasoning task.



Fuzzy ontology: C:\fuzzyOntology.txt

OWL 2 reasoner: HermiT

Output ontology: c:\example.owl

Degrees of truth:
false
0.25
0.5
0.75
true

```
Computing crisp reduction...
  KB with 5 degrees.
  Creating new GCIs...
  Creating new RIAs...
  Mapping ABox...
  Mapping TBox...
  Mapping RBox...
Reduction time: 0.031 seconds.
OWL2 ontology c:/example.owl saved.
```

Fig. 4.10 User interface of DeLorean reasoner

These three technologies show how the expressivity of crisp DL ontologies can be extended using a complementary formalism. In our case, we focused on fuzzy logic, as it provided us with mechanisms to model the uncertainty that is inherent in many of our working scenarios.

4.5 Applying Semantic Web Technologies to Mobile Computing

Advances in mobile computing, with the popularity and widespread and pervasive use of mobile devices and wireless communication technologies, have emphasized the interest in providing mobile users with access to useful information, anywhere and anytime. Besides, in this type of scenarios, it is particularly important to customize the information provided according to the context of the user, to show her only truly relevant data and avoid overloading the user with unneeded information. One of the most important context parameters is the location of the user, which has given rise to the development of a wide variety of LBSs (Ilarri et al. 2010; Schiller and Voisard 2004), such as vehicle tracking applications, friend-finder applications, location-based emergency services, location-based advertisements, and location-based games, among others.

A fundamental building block to define appropriate LBSs for a variety of scenarios is the so-called *location-dependent queries*, which are queries whose answer depends on the location of certain objects/entities (notably, not only the location of the mobile user that submits the query but also other objects that are relevant for the query). These queries thus include *location-dependent constraints*, such as *inside* (that filters the objects that are within a certain area), *nearest* (that retrieves the nearest object or the k-nearest objects of a certain type to a given object), etc. For an in-depth study of location-dependent queries, we refer the interested reader to Ilarri et al. (2010).

As the locations of moving objects (e.g., people, vehicles, etc.) may change at any time, the answer to a location-dependent query may become obsolete in a short time. Therefore, as opposed to traditional queries (*one-shot queries* or *snapshot queries*), location-dependent queries are usually processed as *continuous queries*, that is, as queries whose answer must be updated by the system all the time, until they are explicitly canceled by the user. This implies the need of an efficient approach to keep the answer up-to-date, without the need to reevaluate the whole query from scratch every time that the answer needs to be refreshed. Besides, the answer to a query is usually refreshed periodically: Triggering a query whenever the answer changes is usually not possible, as even if the answer set (i.e., the set of objects that are an answer to the query) does not change, the locations of the objects in the answer set may change all the time and the user is usually interested in their current locations, that therefore will need to be refreshed continually.

It is important to highlight that sometimes a GPS location granularity for location-dependent queries may be unnecessary or even inconvenient. So, the user should be allowed to express locations and location constraints in terms of the location terminology that she requires (the locations could be not only GPS locations but also cities, rooms, buildings, provinces, countries, etc.), which are called *location granules* in Ilarri et al. (2011a). Its use may have an impact on the semantics of the query, on the performance of the query processing, and also on the way the query results are shown to the user. Moreover, location granules should not be considered just a set of geographic locations, as they may also have other features and additional meaning attached; for this reason, we have later proposed the concept of *semantic location granule* (to be described in the next section).

Location-dependent queries would benefit from the semantic management of information about moving objects and scenarios. Acquiring the required knowledge about the existing types of objects and their features, the application scenarios, etc., should not be the responsibility of the user. This has motivated our ongoing work on the SHERLOCK system (to be described later), where we also try to go beyond simple location-dependent queries to more generic *location-dependent requests*, which may be traditional location-dependent queries but also commands requesting objects to perform specific actions (e.g., using their sensors to measure certain values).

Semantic Location Granules

As we have mentioned in the previous section, location-dependent queries are one of the most active matters of study in LBSs. For example, if we are visiting New York and take with us a smartphone, we could query "what are the museums in 1 kilometer?" In the literature, most of the works (Ding et al. 2008; Gedik and Liu 2006; Ilarri et al. 2010; Mokbel et al. 2005; Sistla et al. 1997) consider that the answer to this query is a set of GPS points (that could be located on a map). But in many cases, we do not need the precise GPS location. Instead, we need to use a more abstract notion of location, *area name*, since the geographic coordinates are probably meaningless to us. For instance, if a user wants to go by subway to the MoMA museum in New York, she will not be interested in the precise coordinates of the area where she has to leave the subway, but she may want to know the name of the station near the MoMA museum to leave the subway. It makes no sense to show her a map with a point to leave the subway. She needs the name of the station. To take these situations into account, the concept of *location granule* arises as a set of physical locations (Bernad et al. 2013; Bobed et al. 2011; Ilarri et al. 2007, 2009, 2011). This concept is similar to the concept of *place* in Hightower (2003) and Hoareau and Satoh (2007, 2009) or *spatial granule* in Belussi et al. (2009). Furthermore, we can group a set of location granules under a name to obtain what it is known as a *location granules map*. For example, Madrid could be a location granule of the location granules map of the provinces of Spain.

Although location granules enhance the expressivity of location-dependent queries and this is an important step forward, they are basically a set of GPS points with a name. When we group a set of locations and give them a name, we are implicitly giving them also a meaning. For example, the set of locations that compose Madrid, that is, the location granule with the name Madrid, becomes a city, the capital of Spain. Thus, the location granule Madrid can be seen as a more abstract concept that represents a city or a capital of a country. Implicitly, a location granule has a semantics (it is a city, a country, etc.). To model the semantics of location granules, the concept of *semantic location granule* is introduced in Bernad et al. (2013) and Bobed et al. (2010).

In addition, it is also interesting to consider not only the semantics of the location granules (Madrid is a city, Hyde Park is a park, Spain is a country) but also the semantics of the topological relations between them: For instance, Madrid *is contained* in Spain, or Oxford Street *is adjacent* to Hyde Park. And what is more important is that, if we introduce semantics in location granules and in its topological relations, we can infer implicit knowledge automatically. For instance, if a user is in the city of Zaragoza, it could be inferred that she is also close to France, a country with famous red wines. The interest of linking LBSs and semantics has been emphasized in Ilarri et al. (2011b).

Several works in the field of geographic information systems (GIS; Rigaux et al. 2002; Shekhar and Chawla 2002) have used the relational data model to represent topological relations (e.g., the region connection calculus, RCC; Randell et al. 1992). However, to support reasoning with geographic elements, several previous works have studied the introduction of ontologies in the area of GIS (Couclelis 2010; Lutz and Klien 2006), and the introduction of different types of topological relations in DLs (Haarslev et al. 1998; Lutz and Möller 1997). Despite these efforts, more research is needed in this area to effectively enhance the processing of location-dependent queries with inference capabilities over location granules. For example, *isContained* is a key topological relationship (it allows representing the geographic hierarchy of areas), but the existing proposals to represent such a relationship have important disadvantages (depending on the specific proposal, wrong conclusions may be obtained or a manual assertion of many of the *isContained* relations may be required).

Summing up, the three main goals that are pursued with semantic location granules are:

- To represent the semantics of a set of locations, i.e., to represent the semantics behind location granules.
- To represent the semantics of topological relations between location granules.
- To be able to infer implicit knowledge automatically.

We have proposed two complementary models for semantic location granules (Bobed et al. 2010; Bernad et al. 2013). The former one considers the location granules as instances of a concept *Granule*, and it is oriented to exploit the ABox extending the query model using logical rules; for further details, see Bobed et al. (2010). In this chapter, we will focus on the latter one, which considers that the granules themselves are concepts, as they subsume a set of locations (which now become the instances). As we will see, this latter model allows the DL reasoner to make intensive use of the TBox to infer the containment relationships (Bernad et al. 2013). In the following, we will say granule instead of location granule, for the sake of simplicity.

Modeling Semantic Location Granules as Concepts Let us now discuss modeling semantic location granules as concepts. In this approach, we will consider that a semantic location granule is a concept in a TBox $\mathcal{T}$ of a knowledge representation $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For example, we will consider that *Madrid* and *Spain* are concepts in $\mathcal{T}$. The most straightforward way to express that Madrid is contained in Spain is to add an axiom in $\mathcal{T}$, *Madrid* $\sqsubseteq$ *Spain*, but it fails as we will see in the following. Recall that one of the objectives to introduce semantic location granules is the possibility to express that Madrid is a city or Spain is a country. Again, the most natural way to express these situations is to consider the concepts *City* and *Country,* and add in $\mathcal{T}$ the axioms *Madrid* $\sqsubseteq$ *City* and *Spain* $\sqsubseteq$ *Country*. But if we want to say that a city cannot be a country, that is, *City* $\sqsubseteq \neg$ *Country*, then the T Box $\mathcal{T}$ becomes inconsistent: It is inferred that Madrid is a city and a country, *Madrid* $\sqsubseteq$ *Spain* $\sqsubseteq$ *Country*, which is a contradiction. With this simple example, we can see that it is not so trivial to express the content topological relationship using a DL. The main problem is that the subsumption operator ($\sqsubseteq$) is used to express that Madrid is an area inside the area of Spain, and to express that Madrid is a city, that is, the subsumption operator is used to express the content relationship between areas as well as the *is a* relationship between concepts.

To avoid the above problem, in the formalization of semantic granules and semantic granule maps with DLs, we use a transitive role, named *isContained*, and roles $loc_{X_1}, \ldots, loc_{X_n}$. Intuitively, the role *isContained* is used to express that a granule is geographically contained in another one by subsumption and participation in the relationship *isContained* (e.g., *NewYork* $\sqsubseteq \exists$*isContained.EEUU*); and $loc_{X_1}, \ldots, loc_{X_n}$ are the coordinates of a point. These concrete features allow us to define areas; for example, $loc_x \geq 10 \sqcap loc_x \leq 20 \sqcap loc_y \geq 10 \sqcap loc_y \leq 30$ represents a rectangle.

Definition 1 An area concept $f(loc_{X_1}, \ldots, loc_{X_n})$ is a concept built with the constructors $\sqcup$ and $\sqcap$, and the roles $loc_{X_1}, \ldots, loc_{X_n}$. An area concept name A is a concept name such that $A \equiv f(loc_{X_1}, \ldots, loc_{X_n})$, where f is an area concept. The set of names of area concepts is denoted by N_A.

For example, $\exists loc_x \geq 10 \sqcap \exists loc_x \leq 20 \sqcap \exists loc_y \geq 10 \sqcap \exists loc_y \leq 30$ is an area concept, while $\exists loc_x \geq 10 \sqcap \exists loc_x \leq 20 \sqcap \exists loc_y \geq 10 \sqcap \exists loc_y \leq 30 \sqcap City$ is not.

Now, we formalize the definition of semantic location granule and semantic granules map.

Definition 2 *Given* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *a knowledge representation and a granule map* $M = \{G_1, \ldots, G_n\}$ *with* G_i *granules, a semantic granule map is a tuple* (*M, $\mathcal{K}$, area, semGranule*) *where area and semGranule are functions from the set of granules of* M *to the concept names of* $\mathcal{T}$*; that is, area, semGranule* $M_G \rightarrow N_C$*, such that the following must be satisfied for all* $G \in M_G$.

1. $semGranule\ (G) \sqsubseteq area\ (G)$
2. $area\ (G) \sqsubseteq \exists\ isContained.semGranule(G)$

The concept names $semGranules(G)$ are called semantic granules.

Let us show an example to explain the definition. Let M be a granule map with location granules $M = \{ZaragozaGr, AragonGr, MadridGr, SpainGr\}$, and $\mathcal{T}$ the TBox of a knowledge representation $\mathcal{K}$ defined in Table 4.4 (to keep explanations easier to follow, we represent geographic areas in the TBox by simple rectangles instead of the real geographic limits).

We define a semantic granule map (*M, $\mathcal{K}$, area, semGranule*), where *area* and *semGranule* are functions *area* (*SpainGr*)=*SpainArea*, etc., and *semGranule* (*SpainGr*)=*Spain*, etc. We can ensure that this is a semantic granule map since it holds the conditions (1) and (2) of Definition 2 from axioms (5)–(8), and (9)–(12), respectively. Figure 4.11 shows the map corresponding to the modeled area.

Intuitively, the condition (1) of Definition 2 says that a semantic granule is not only its geographic area but it could also have more attributes. For example, *Zaragoza* is an area and a *City*, and *Spain* is an area and a *Country*. The condition (2) allows to establish qualitative relations between granules such as "Zaragoza is a city in Spain," i.e., $Zaragoza \sqsubseteq \exists\ isContained.Spain$, or to express the concept "Aragon's wines," $AragonWine \equiv Wine \sqsubseteq \exists\ isContained.Aragon$. Note that we do not express the concept Aragon's wine as $Wine \sqcap Aragon$, since *Aragon* is a *Region* and wines are not regions; and similarly, Aragon's wines are not defined as $Wine \sqcap AragonArea$, as wines could not have location information. We have

Table 4.4 TBox axioms involving semantic granules

Axiom	Definition
(1)	$ZaragozaArea \equiv \exists loc_x \leq 25 \sqcap \exists loc_x \leq 30 \sqcap \exists loc_y \geq 23 \sqcap \exists loc_y \leq 30$
(2)	$AragonArea \equiv \exists loc_x \geq 25 \sqcap \exists loc_x \leq 30 \sqcap \exists loc_y \geq 20 \sqcap \exists loc_y \leq 32$
(3)	$MadridArea \equiv loc_x \geq 15 \sqcap \exists loc_x \leq 20 \sqcap \exists loc_y \geq 17 \sqcap \exists loc_y \leq 23$
(4)	$SpainArea \equiv \exists loc_x \geq 5 \sqcap \exists loc_x \leq 35 \sqcap \exists loc_y \geq 0 \sqcap \exists loc_y \leq 35$
(5)	$Zaragoza \equiv ZaragozaArea \sqcap City$
(6)	$Aragon \equiv AragonArea \sqcap Region$
(7)	$Madrid \equiv MadridArea \sqcap City$
(8)	$Spain \equiv SpainArea \sqcap Country$
(9)	$ZaragozaArea \sqsubseteq \exists\ isContained.Zaragoza$
(10)	$AragonArea \sqsubseteq \exists\ isContained.Aragon$
(11)	$MadridArea \sqsubseteq \exists\ isContained.Madrid$
(12)	$SpainArea \sqsubseteq \exists\ isContained.Spain$
(13)	*Region, Country, City* are mutually disjoint
(14)	$RedWine \sqsubseteq Wine$
(15)	$AragonWine \equiv Wine \sqcap \exists\ isContained.Aragon$

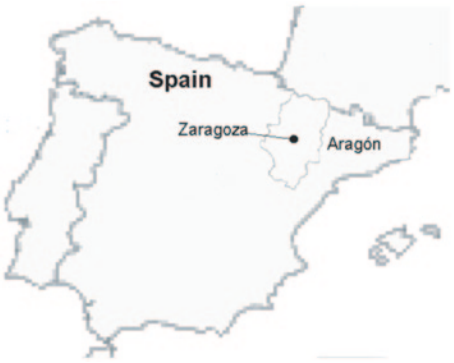


Fig. 4.11 Sample granule map

divided the containment relationship in two parts: One to make calculations about areas (quantitative reasoning) using the subsumption relationship, and another one to establish relationships with other concepts (qualitative reasoning) using the *is-Contained* relationship.

From this model, a DL reasoner can deduce a number of facts, as we explain in the following:

Proposition 1 *A reasoner under conditions of Definition 2 can infer that:*

1. *A granule G is contained in a granule G', i.e., it can be deduced that* $semGranule(G) \sqsubseteq \exists\ isContained.semGranule(G')$
2. *A granule G intersects a granule G'*

In the example above, it can be inferred that Zaragoza is contained in Spain, $Zaragoza \sqsubseteq ZaragozaArea \sqsubseteq SpainArea \sqsubseteq \exists\ isContained.Spain$. The second statement is obvious since it is equivalent to asking if the area concept $area\ (G) \sqcap area\ (G')$ is satisfiable.

Another interesting remark is that the content of a granule only depends on its area. What does this mean? For example, let us suppose that we define two different semantic granules with the same area, *VaticanCity* and *VaticanCountry*, defined as *VaticanArea* ⊓ *City* and *VaticanArea* ⊓ *Country,* respectively. We would like the content of Vatican as a city (∃ *isContained. VaticanCity*) to be equal to the content of the Vatican as a country (∃ *isContained. VaticanCountry*), even when a country is not a city. Due to the conditions (1) and (2) of Definition 2 and the transitivity of the role *isContained,* we can conclude that in our model ∃ *isContained.VaticanCity* ≡ ∃ *isContained.VaticanCountry* as it is shown in the following proposition.

Proposition 2 Let *G be a granule under conditions of Definition 2. Then it holds that*:
$\exists\ isContained.semGranule\ (G) \equiv \exists\ isContained.area(G)$

and therefore*, if* G_1 *and* G_2 *are granules such that area* $(G_1) \equiv$ *area* (G_2), *then* $\exists\ isContained.semGranule\ (G_1) \equiv \exists\ isContained.semGranule\ (G_2)$

For further details on the model and the proofs of the different propositions, we refer the interested reader to Bernad et al. (2013).

Semantic Management of LBSs: SHERLOCK

The astonishing penetration of mobile computing in our daily lives, thanks to devices such as smartphones and tablets, leads us to a scenario where mobile users have access to huge amounts of information *anytime* and *anywhere*. Thousands of applications (also known as *apps*) for their smart devices are available to offer them information about transportation, entertainment, culture, etc. The Web has also been growing steadily in the last few years with tons of potentially useful information. Therefore, users are starting to be overwhelmed with the amount of data they receive from different sources, as it is sometimes difficult for people to distinguish which information is valuable.

For example, imagine a researcher attending a conference who arrives on an evening flight and needs to reach her city hotel. At first, she would be interested in transport information and she might need to know the different options (e.g., buses, metros, taxis, or car rental options), traffic conditions, and perhaps even where available parking spaces are located. This information could be obtained by visiting a tourist office, searching a local transportation web site, or even downloading a mobile app. After checking in, she could be interested in finding other nearby conference attendees to talk to them or even to go sightseeing (again, she should browse the Web to find information about interesting places to visit). Thus, it is the user herself who is in charge of knowing/finding the interesting and updated information sources and gathering and correlating all this information; even worse, she will have to know/find all these *updated* information about each city she visits.

Semantic Web techniques become particularly useful in these scenarios. First, they can be used to understand the information needs of users controlling the ambiguities of natural language (as already explained in this chapter). Also, these techniques can help to find the most appropriate information from a range of different sources by inferring useful information providers. Finally, information extracted from heterogeneous sources can be presented in an integrated way by using common representation models such as ontologies.

Overview We present SHERLOCK (System for Heterogeneous mobilE Requests by Leveraging Ontological and Contextual Knowledge) (Yus et al. 2014a), a system to provide mobile users with interesting LBSs. SHERLOCK (http://sid.cps.unizar.es/SHERLOCK) relieves users from the need to obtain up-to-date information about the services they need. Using ontologies to model the knowledge related to these services, SHERLOCK devices exchange information among themselves, for example, about LBSs in the area. Also, with the help of a semantic reasoner, our system is able to determine which information could be interesting for a user regarding her context, and to obtain this information from objects nearby by leveraging the collaboration among devices. For that purpose, SHERLOCK deploys a network or mobile agents (Lange and Oshima 1999; Spyrou et al. 2004; Trillo et al. 2007b) which move from one device to another autonomously to be near the needed information source and collect data (see Fig. 4.12).

Obtaining Knowledge from Devices Around SHERLOCK is based on knowledge sharing among devices. Each participating device starts with a basic OWL local ontology containing the basic terms to define LBSs, with concepts such as “Service,” “Provider,” “Parameter” (see Fig. 4.13 where the basic terms are in bold font). SHERLOCK devices learn from the interaction with others: When two devices meet, they share part of their local ontologies. A SHERLOCK device that receives new knowledge integrates it with its local ontology, and thus it can use this new knowledge to provide the user with more interesting information. For example, the device of a user that lives in Zaragoza (Spain) knows transportation concepts such as “Taxi” or “"Bus”; if the user travels to Thrissur (India) and her device starts

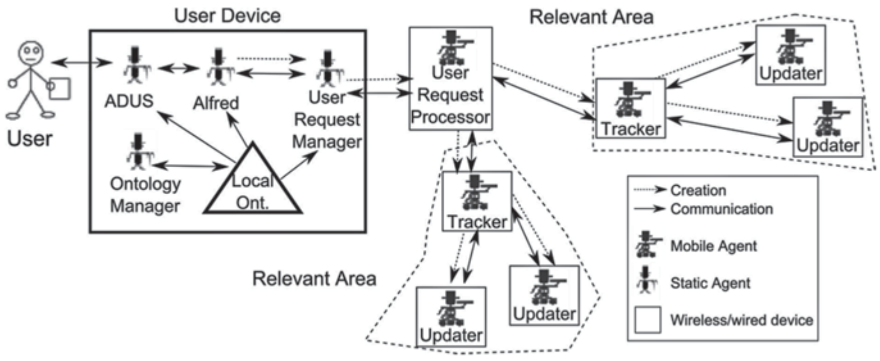


Fig. 4.12 Agent network deployed to process a request

communicating with others, it can learn that "Tuk-Tuk" (a vehicle defined in the ontology as a private transport that carries people to a certain destination) is similar to a taxi, and thus, it can be interesting for a user that needs transport.

In SHERLOCK, there is an (static) agent in charge of managing the local ontology of the device. This agent, called *Ontology Manager* (OM), has two main tasks: (1) sharing knowledge with OMs in other devices, and (2) integrating the knowledge received. OMs are continuously asking for knowledge related to the context of the user to new devices discovered. Also, if the user shows her interest in a specific location (e.g., downtown) or concept (e.g., hotel), OMs broadcast a message asking for knowledge related to it. An OM that receives a knowledge petition applies ontology modularization techniques (Stuckenschmidt et al. 2009) to extract relevant knowledge from its local ontology. When the new knowledge is received, OMs

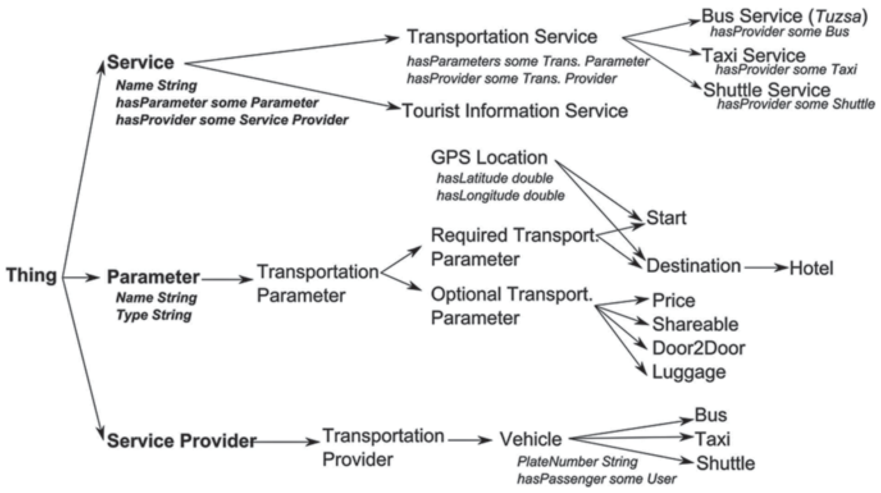


Fig. 4.13 Subset of an ontology that defines an LBS: "Transportation Service"

apply ontology matching techniques (Euzenat and Shvaiko 2007) to integrate the new knowledge into its local ontology. In this way, an OM will discover semantic relationships among terms such as synonymy (e.g., the terms "Taxi" and "Cab" are synonyms) or hypernymy (e.g., the term "Car" is a hypernym of the term "Taxi").

Generating an Information Request When the user shows her interest in a certain location, SHERLOCK on her device uses the knowledge in its local ontology to help her generate an information request. The goal of this step is to generate a request that SHERLOCK will later process to find the information that the user needs.

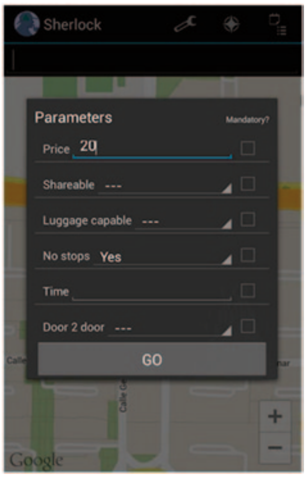
There are three (static) agents involved in this process: *ADUS, Alfred*, and *User Request Manager* (URM). ADUS is in charge of generating graphical interfaces appropriate to show information to the user and get her input. Alfred is in charge of storing information about the user and her device (e.g., user preferences, previous requests, and technical capabilities of the device). This information is especially relevant to infer interesting information from previous user requests when generating a new request. For example, Alfred can infer that the user usually selects taxis and shuttles when looking for transportation, and thus, she seems more interested in private transportation than in public transportation. Finally, URMs interact with the user to generate the final information request. Using the location that the user selected, the URM infers which LBSs are related to it. This is done by finding all the LBSs defined in the ontology that are related (e.g., through a property) to the type of location selected. For example, if the user selects a hotel, the URM will infer that services to book a room or to find a transport to go there can be interesting. The context of the user is also used to filter out some of these services (e.g., if the user is already in a taxi, the service to find transportation would probably not be interesting for her). Then, the URM presents the possible services to the user and when she selects one of them, the URM presents its parameters (if any) to find the most appropriate service provider for the selected service. Some of these parameters will be filled in by the URM using the information stored by Alfred.

Processing an Information Request Once SHERLOCK has generated an information request to find the most appropriate service providers for the user, the next step is processing it. Following the approach presented in LOQOMOTION (Ilarri et al. 2006b) to process location-dependent queries, SHERLOCK creates a network of mobile agents to find these service providers. These mobile agents, used as a way to balance the computing load and minimize the network latency, consider every device in the scenario as a potential processing node. Mobile agents continuously evaluate the appropriateness of the current device where they are executing for the task they are performing. As a result, these agents can stay in the same device, move to another one where the performance is expected to be better, or even create new helper mobile agents if they cannot solve the situation alone.

There are three types of mobile agents involved in the processing of an information request: *User Request Processor* (URP), *Trackers*, and *Updaters*. URPs are created to continuously process the user information request and return the results to their URMs (that will present these results to the user). URPs have to define the geographic area that would be interesting to obtain information related to the user request. Finally, a URP creates Tracker agents to monitor the selected area(s). A Tracker is in charge of monitoring an area to find relevant objects that can provide the service that the user needs. To achieve this goal, Trackers create a network of Updater agents and correlate their results. The Tracker is responsible for maintaining the network of Updaters trying to cover the area completely and with an adequate frequency (due to communication delays). Finally, Updaters are in charge of communicating with objects asking for the ontological context that describes them. Updaters are provided by their Trackers with the ontological description of the service providers interesting for the user. So, if an object's profile fulfills this description, then its information (especially its location) will be returned to the Tracker. Note that Updaters can use the reasoner on the device they are executing to check if an unknown object could be classified as an instance of the target provider it is looking for.

Prototype We have developed a prototype of the SHERLOCK (Yus et al. 2013b) as an Android app (see Figs. 4.14–4.16 for some screenshots of the prototype). The prototype uses the OWL API (Horridge and Bechhofer 2011), an ontology API to manage OWL 2 ontologies in Java applications, the JFact reasoner, and the SPRINGS mobile agent platform (Ilarri et al. 2006a). Using Semantic Web technologies (ontology APIs and DL reasoners) on current mobile devices are feasible, as we studied in Yus et al. (2013a) and Bobed et al. (2014). In addition, in Bobed et al. (2014), we evaluated the performance of semantic reasoners on smart devices and our results show that current smartphones can handle reasoning on small/medium ontologies.

First, the SHERLOCK prototype asks the user for some information such as her name and "profile" (e.g., person, researcher, taxi, or bus). Then, the prototype creates P2P networks using WiFi to communicate with other devices and to exchange OWL ontologies. Finally, the prototype helps users to create their information re-



Parameters Mandatory?

Price 20
Shareable ---
Luggage capable ---
No stops Yes
Time
Door 2 door ---

Fig. 4.14 The user fills in parameters to select the most appropriate service provider

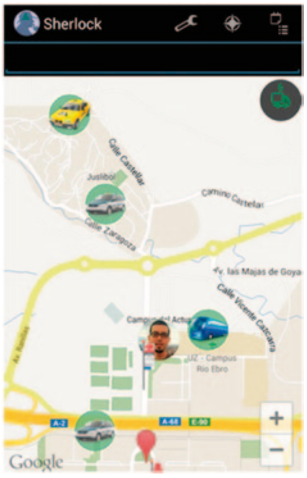


Fig. 4.15 Real-time location of service providers are shown on a map

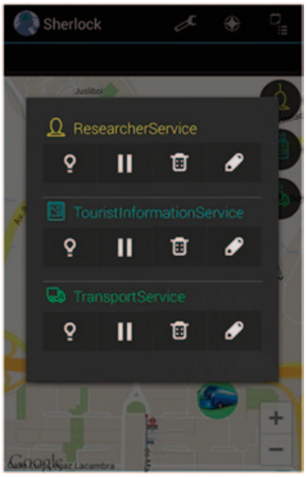


Fig. 4.16 Different information requests can be processed for a user

quests and to find other SHERLOCK devices around whose "profile" matches with the kind of appropriate service providers for the user (Fig. 4.15).

4.6 Discussion

Throughout this chapter, we have presented several examples of how the use of semantic techniques leads to the development of smarter information systems. Making the computer aware of the semantics of the data will require still a long road of research in many fields. However, as we have shown, Semantic Web technologies in their current state make it possible to improve existing approaches, and to devise new and more intelligent applications.

In the SID research group, we are following this line of thought and research, attempting always to go a step further into exploiting semantics to improve the capabilities of our systems. We have seen how the use of semantic techniques has allowed us to improve Web searches, to perform semantic keyword-based searches on heterogeneous information systems, or to develop a platform to provide smarter LBSs, for example. However, our lines of research, which are directly implemented in our prototypes, are still open. Among others, some of the issues we are currently working on are:

- There is plenty of work to be done yet in the field of obtaining the semantics out from plain keywords and plain text. We are studying how to combine ontologies with NLP techniques to enhance this process, and solve many of the linguistic problems that exist (e.g., ambiguity and multilingualism). Moreover, with the know-how acquired during the development of QueryGen, Doctopush, and GENIE, we also want to study the open problem of Question Answering, which has recently attracted new attention (Lopez et al. 2011, 2013) due to the possibilities that ontologies provide to interpret the meaning of the posed queries.
- Regarding semantic searches and semantic LBSs, we want to study the inclusion of context information. User preferences could be introduced in our systems to provide better results in terms of more intelligent services providing more relevant results. For example, in some cases, some data could be provided to the user even in the absence of explicit queries, such as in *mobile recommendations* (Rodríguez-Hernández and Ilarri 2014). As another example, if we allow queries such as "retrieve hotels that are near," the notion of *near* is imprecise, and depends on the context (e.g., is the user walking or driving?) and the user preferences.
- In the context of location modeling, we also want to explore further the modeling capabilities of DLs to capture different spatial relationships and enhance our semantic location model. We will analyze how to introduce RCC relations other than *isContained* (e.g., inner and outer tangential relations), and how to model the dependency between all the RCC relationships. Our objective is to model RCC in a DL in such a way that, if the topology is possible, then the TBox with the axioms describing the topology is consistent and all topology relationships that we can derive from the TBox are realizable.
- Finally, we plan to study the semantic management of multimedia data in the context of SHERLOCK. Current mobile devices generate large amounts of photos and videos that SHERLOCK could take into account when offering the user with interesting information. We will explore the integration of our previous experience on multimedia information management (Yus et al. 2014b) with the semantic management of data of SHERLOCK.

The Semantic Web and its associated technologies are here to stay and are no longer restricted to web environments. Of course, there are still open issues and a long road to research in this field. For example, there is still no general purpose search engine that, given a query, returns the exact answer the user is looking for. For instance, a user that inputs the query "what is the meaning of life the universe and everything?" would expect the result to be just "42" and not thousands of web sites talking about The Hitchhiker's Guide to the Galaxy. Although some important knowledge bases have been generated in the last few years that might contain this information, we are still far from a scenario where computers understand the meanings behind any type of data and data repository. Nevertheless, in parallel with the enhancement of the current semantic techniques and Semantic Web technologies, we have shown that we can embrace and integrate them to change the way we devise applications.

Acknowledgment This work has been supported by the CICYT projects TIN2010-21387-C02-02 and TIN2013-46238-C4-4-R, and DGA-FSE.

References

Aho, A., Sethi, R., & Ullman, J. (2006). *Compilers: Principles, techniques, and tools*. New York: Addison-Wesley.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Scheneider, P. (2003). *The description logic handbook: Theory, implementation and applications*. New York: Cambridge University Press.

Banerjee, S., & Pedersen, T. (2003). *Extended gloss overlaps as a measure of semantic relatedness*. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03) (pp. 805–810). Acapulco, Mexico: Morgan Kaufmann.

Belussi, A., Combi, C., & Pozzani, G. (2009). *Formal and conceptual modeling of Spatio-temporal granularities*. In Proceedings of the 13th International Database Engineering & Applications Symposium (IDEAS'09) (pp. 275–283). Calabria, Italy: ACM.

Bernad, J., Bobed, C., Mena, E., & Ilarri, S. (2013). A formalization for semantic location granules. *International Journal of Geographical Information Science, 27*(6), 1090–1108.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 34–43.

Bird, S., Day, D., Garofolo, J. S., Henderson, J., Laprun, C., & Liberman, M. (2000). *ATLAS: A flexible and extensible architecture for linguistic annotation*. In Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC'00) (pp. 1699–1706).

Bizer, C., Heath, T., & Berners-Lee, T. (2009a). Linked Data—the story so far. *International Journal on Semantic Web and Information Systems, 5*(3), 1–22.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009b). DBpedia—a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web, 7*(3), 154–165.

Bobed, C. (2013). *Semantic keyword-based search on heterogeneous information systems*. PhD in Computer Science, University of Zaragoza, Spain.

Bobed, C., Ilarri, S., & Mena, E. (2010). *Exploiting the semantics of location granules in location-dependent queries*. In Proceedings of the 14th East-European Conference on Advances in Databases and Information Systems (ADBIS'10) (Vol. 6295, pp. 79–93). Berlin: Springer.

Bobed, C., Bobillo, F., Yus, R., Esteban, G., & Mena, E. (2014). *Android went semantic: Time for evaluation*. In Proceedings of the 3rd International Workshop on OWL Reasoner Evaluation (ORE'14) (pp. 23–29). CEURWS.

Bobillo, F. (2008). *Managing vagueness in ontologies*. PhD in Computer Science, University of Granada, Spain.

Bobillo, F., & Straccia, U. (2008). *fuzzyDL: An expressive fuzzy description logic reasoner*. In Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008) (pp. 923–930). IEEE Computer Society.

Bobillo, F., & Straccia, U. (2011). Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning, 52*(7), 1073–1094.

Bobillo, F., & Straccia, U. (2012). Generalized fuzzy rough description logics. *Information Sciences, 189*(1), 43–62.

Bobillo, F., & Straccia, U. (2013). Aggregation operators for fuzzy ontologies. *Applied Soft Computing, 13*(9), 3816–3830.

Bobillo, F., Delgado, M., & Gómez-Romero, J. (2012a). DeLorean: A reasoner for fuzzy OWL 2. *Expert Systems with Applications, 39*(1), 258–272.

Bobillo, F., Delgado, M., Gómez-Romero, J., & Straccia, U. (2012b). Joining Gödel and Zadeh fuzzy logics in fuzzy description logics. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 20*(4), 475–508.

Couclelis, H. (2010). Ontologies of geographic information. *International Journal of Geographical Information Science, 24*(12), 1785–1809.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). *A framework and graphical development environment for robust NLP tools and applications*. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02) (pp. 168–175). Philadelphia: ACL.

d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., & Motta, E. (2007). *Characterizing knowledge on the semantic web with Watson*. In Proceedings of the 5th International Workshop on Evaluation of Ontologies and Ontology-Based Tools (EON'07) (Vol. 329, pp. 1–10). Busan: CEUR–WS.

Ding, H., Trajcevski, G., & Scheuermann, P. (2008). Efficient maintenance of continuous queries for trajectories. *Geoinformatica, 12*(3), 255–288.

Euzenat, J., & Shvaiko, P. (2007). *Ontology matching* (Vol. 18). Berlin: Springer.

Frank, A. (2003). *Chapter 2: Spatio-temporal databases*. Berlin: Springer Verlag.

Fu, H., & Anyanwu, K. (2011). *Effectively interpreting keyword queries on RDF databases with a rear view*. In Proceedings of the 10th International Semantic Web Conference (ISWC'11) (Vol. 7031, pp. 193–208). Springer.

Garrido, A. L., & Ilarri, S. (2014). *TMR: A semantic recommender system using topic maps on the items descriptions*. In The Semantic Web: ESWC 2014 Satellite Events (Vol. 8798, pp. 213–217). Springer.

Garrido, A. L., Gómez, O., Ilarri, S., & Mena, E. (2011). *NASS: News annotation semantic system*. In Proceedings of the 23rd IEEE International Conference on Tools With Artificial Intelligence (ICTAI'11) (pp. 904–905). IEEE Computer Society.

Garrido, A. L., Gómez, O., Ilarri, S., & Mena, E. (2012). *An experience developing a semantic annotation system in a media group*. In Proceedings of the 17th International Conference on Applications of Natural Language Processing to Information Systems (NLDB'12) (pp. 333–338). Springer.

Garrido, A. L., Granados-Buey, M., Ilarri, S., & Mena, E. (2013a). *GEO-NASS: A semantic tagging experience from geographical data on the media*. In Proceedings of the 17th East-European Conference on Advances in Databases and Information Systems (ADBIS'13) (pp. 56–69). Springer.

Garrido, A. L., Granados-Buey, M., Escudero, S., Ilarri, S., Mena, E., & Silveira, S. (2013b). *TM-Gen: A topic map generator from text documents*. In Proceedings of the 25th IEEE International Conference on Tools With Artificial Intelligence (ICTAI'13) (pp. 735–740). IEEE Computer Society.

Garrido, A. L., Peiro, A., & Ilarri, S. (2014a). *Hypatia: An expert system proposal for documentation departments*. In Proceedings of the 12th IEEE International Symposium on Intelligent Systems and Informatics (SISY'14) (pp. 315–320). IEEE Computer Society.

Garrido, A. L., Pera, M. S., & Ilarri, S. (2014b). *SOLER, a semantic and linguistic approach for book recommendations*. In Proceedings of the 14th IEEE International Conference on Advanced Learning Technologies (ICALT'14) (pp. 524–528). IEEE Computer Society.

Garrido, A. L., Granados-Buey, M., Escudero, S., Peiro, A., Ilarri, S., & Mena, E. (2014c). *The GENIE project—a semantic pipeline for automatic document categorisation*. In Proceedings of the 10th International Conference on Web Information Systems and Technologies (WEBIST'14) (pp. 161–171). SCITEPRESS.

Gedik, B., & Liu, L. (2006). MobiEyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing, 5*(10), 1384–1402.

Glimm, B., Horrocks, I., Motik, B., Stoilos, G., & Wang, Z. (2014). HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning, 40*(2–3), 89–116.

Gómez-Pérez, A., Fernández-López, M., & Corcho, Ó. (2004). *Ontological engineering*. London: Springer.

Gracia, J., & Mena, E. (2008, Sept). *Web-based measure of semantic relatedness*. In Proceedings of the 9th International Conference on Web Information Systems Engineering (WISE'08) (Vol. 5175, pp. 136–150). Springer.

Gracia, J., & Mena, E. (2009). *Multiontology semantic disambiguation in unstructured web contexts*. In Proceedings of Workshop on Collective Knowledge Capturing And Representation (CKCaR'09).

Gracia, J., d'Aquin, M., & Mena, E. (2009). *Large scale integration of senses for the semantic web*. In Proceedings of the 18th International World Wide Web Conference (WWW'09) (pp. 611–620). ACM.

Granados-Buey, M., Garrido, A. L., Escudero, S., Trillo, R., Ilarri, S., & Mena, E. (2014a). *SQX-Lib: Developing a semantic query expansion system in a media group*. In Proceedings of the 36th European Conference on IR Research (ECIR'14) (Vol. 8416, pp. 780–783). Springer.

Granados-Buey, M., Garrido, A. L., & Ilarri, S. (2014b). *An approach for automatic query expansion based on NLP and semantics*. In Proceedings of the 25th International Conference on Database and Expert Systems Applications (DEXA'14) (pp. 349–356). Springer.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*(2), 199–220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies, 43*(5–6), 907–928.

Haarslev, V., Lutz, C., & Möller, R. (1998). *Foundations of spatioterminological reasoning with description logics*. In Proceedings of the 6th international conference on principles of knowledge representation and reasoning (KR'98) (pp. 112–123). Morgan Kaufmann.

Harris, S., Seaborne, A., & Prud'hommeaux, E. (2013). SPARQL 1.1 Query Language. (http://www.w3.org/TR/sparql11-query). Accessed 20 April 2015.

Hightower, J. (2003). *From position to place*. In Proceedings of the 2003 workshop on location-aware computing (pp. 10–12). Springer.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2012). OWL 2 web ontology language primer (Second Edition). (http://www.w3.org/TR/owl-primer). Accessed 20 April 2015.

Hoareau, C., & Satoh, I. (2007). *A model checking-based approach for location query processing in pervasive computing environments*. In Proceedings of the 2nd International Workshop on Pervasive Systems 2007(PerSys'07) (Vol. 4806, pp. 866–875). Springer.

Hoareau, C., & Satoh, I. (2009). *From model checking to data management in pervasive computing: A location-based query-processing framework*. In Proceedings of the ACM International Conference on Pervasive Services (ICPS'09) (pp. 41–48). ACM.

Horridge, M., & Bechhofer, S. (2011). The OWL API: A java API for OWL ontologies. *Semantic Web, 2*(1), 11–21.

Ilarri, S., Trillo, R., & Mena, E. (2006a). *SPRINGS: A scalable platform for highly mobile agents in distributed computing environments*. In Proceedings of the 4th International WoWMoM 2006 Workshop on Mobile Distributed Computing (MDC'06) (pp. 633–637). IEEE Computer Society.

Ilarri, S., Mena, E., & Illarramendi, A. (2006b). Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions on Mobile Computing, 5*(8), 1029–1043.

Ilarri, S., Mena, E., & Bobed, C. (2007). *Processing location-dependent queries with location granules*. In Proceedings of the 2nd International Workshop on Pervasive Systems 2007(PerSys'07) (Vol. 4806, pp. 856–866). Springer.

Ilarri, S., Corral, A., Bobed, C., & Mena, E. (2009). *Probabilistic granule-based inside and nearest neighbor queries*. In Proceedings of the 13th East-European Conference on Advances in Databases And Information Systems (ADBIS'09) (Vol. 5739, pp. 103–117). Springer.

Ilarri, S., Mena, E., & Illarramendi, A. (2010). Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys, 42*(3), 1–73.

Ilarri, S., Bobed, C., & Mena, E. (2011a). An approach to process continuous location-dependent queries on moving objects with support for location granules. *Journal of Systems and Software, 84*(8), 1327–1350.

Ilarri, S., lllarramendi, A., Mena, E., & Sheth, A. (2011b). Semantics in location-based services—guest editors' introduction for special issue. *IEEE Internet Computing, 15*(6), 10–14.

ISO/IEC. (2011). ISO/IEC 9075:2011 Standard, Information Technology—Database Languages—SQL.

Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features*. In Proceedings of the 10th European conference on machine learning (ECML 1998) (Vol. 1398, pp. 137–142). Springer.

Kaufmann, E., & Bernstein, A. (2010). Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web, 8*(4), 377–393.

Lange, D. B., & Oshima, M. (1999). Seven good reasons for mobile agents. *Communications of the ACM, 42*(3), 88–89.

Lopez, V., Uren, V. S., Sabou, M., & Motta, E. (2011). Is question answering fit for the semantic web? a survey. Semantic Web -Interoperability, Usability. *Applicability, 2*(2), 125–155.

Lopez, V., Unger, C., Cimiano, P., & Motta, E. (2013). Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web, 21*(0), 3–13. (Special Issue on Evaluation of Semantic Technologies).

Lutz, M., & Klien, E. (2006). Ontology-based retrieval of geographic information. *International Journal of Geographical Information Science, 20*(3), 233–260.

Lutz, C., & Möller, R. (1997). *Defined topological relations in description logics*. In Proceedings of the 1997 international workshop on description logics (DL'97) (pp. 27–29). Morgan Kaufmann Publishers Inc.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York: Cambridge University Press.

Mena, E., & Illarramendi, A. (2001). *Ontology-based query processing for global information systems*. Boston: Kluwer.

Mendes, P., Jakob, M., & Bizer, C. (2012). *DBpedia: A multilingual cross-domain knowledge base*. In Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12) (pp. 1813–1817). European language resources association (ELRA).

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM, 38*(11), 39–41.

Mokbel, M. F., Xiong, X., Hammad, M. A., & Aref, W. G. (2005). Continuous query processing of spatio-temporal data streams in PLACE. *Geoinformatica, 9*(4), 343–365.

Motik, B., Shearer, R., & Horrocks, I. (2009). Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research, 36*(1), 165–228.

Pepper, S., & Moore, G. (2001). XML Topic Maps (XTM) 1.0-TopicMaps.org Specification. (http://www.topicmaps.org/xtm). Accessed 20 April 2015.

Po, L. (2009). *Automatic lexical annotation: An effective technique for dynamic data integration*. PhD in Computer Science, Doctorate School of Information and Communication Technologies, University of Modena e Reggio Emilia, Italy.

Po, L., Sorrentino, S., Bergamaschi, S., & Beneventano, D. (2009). *Lexical knowledge extraction: An effective approach to schema and ontology matching*. In Proceedings of the 10th European Conference on Knowledge Management (ECKM'09) (pp. 617–626). Academic Publishing Limited.

Randell, D. A., Cui, Z., & Cohn, A. G. (1992). *A spatial logic based on regions and connection*. In Proceedings of the 3rd International Conference on Principles Of Knowledge Representation and Reasoning (KR'92) (pp. 165–176). Morgan Kaufmann.

Rigaux, P., Scholl, M., & Voisard, A. (2002). *Spatial databases with application to GIS*. San Francisco: Morgan Kaufmann.

Rodríguez-Hernández, M. C., & Ilarri, S. (2014). *Towards a context-aware mobile recommendation architecture*. In Proceedings of the 11th International Conference on Mobile Web Information Systems (MobiWIS'14) (pp. 56–70). Springer.

Russell, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach*. Englewood Cliffs: Prentice-Hall.

Schiller, J., & Voisard, A. (2004). *Location-based services*. San Francisco: Morgan Kaufmann.

Sekine, S., & Ranchod, E. (Eds.). (2009). *Named entities: Recognition, classification and use*. Amsterdam: John Benjamins.

Shadbolt, N., Hall, W., & Berners-Lee, T. (2006). The semantic web revisited. *IEEE Intelligent Systems, 21*(3), 96–101.

Shekhar, S., & Chawla, S. (2002). *Spatial databases: A tour*. Upper Saddle River: Prentice Hall.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web, 5*(2), 51–53.

Sistla, A. P., Wolfson, O., Chamberlain, S., & Dao, S. (1997). *Modeling and querying moving objects*. In Proceedings of the 13th International Conference on Data Engineering (ICDE'97) (pp. 422–432). IEEE Computer Society.

Smeaton, A. F. (1999). Using NLP or NLP resources for information retrieval tasks. In T. Strzalkowski (Ed.), *Natural language information retrieval* (Vol. 7, pp. 99–111). Dordrecht: Springer.

Spyrou, C., Samaras, G., Pitoura, E., & Evripidou, P. (2004). Mobile agents for wireless computing: The convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications, 9*(5), 517–528.

Stuckenschmidt, H., Parent, C., & Spaccapietra, S. (2009). *Modular ontologies: Concepts, theories and techniques for knowledge modularizatio*n (Vol. 5445). New York: Springer.

Trillo, R., Gracia, J., Espinoza, M., & Mena, E. (2007a). Discovering the semantics of user keywords. *Journal on Universal Computer Science, 13*(12), 1908–1935.

Trillo, R., Ilarri, S., & Mena, E. (2007b). *Comparison and performance evaluation of mobile agent platforms*. In Proceedings of the 3rd International Conference on Autonomic and Autonomous Systems (ICAS'07) (pp. 41–46). IEEE Computer Society.

Trillo, R., Po, L., Ilarri, S., Bergamaschi, S., & Mena, E. (2011). Using semantic techniques to access web data. *Information Systems, 36*(2), 117–133. (Special Issue: Semantic Integration of Data, Multimedia, and Services).

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). *Ontology-based integration of information—a survey of existing approaches*. In Proceedings of the IJCAI Workshop: Ontologies and Information Sharing (pp. 108–117). CEUR–WS.

Yus, R., Bobed, C., Esteban, G., Bobillo, F., & Mena, E. (2013a). *Android goes Semantic: DL reasoners on smartphones*. In Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE'13) (pp. 46–52). CEUR–WS.

Yus, R., Mena, E., Ilarri, S., & Illarramendi, A. (2013b). *SHERLOCK: A system for location-based services in wireless environments using semantics*. In Proceedings of the 22nd International World Wide Web Conference (WWW'13) (pp. 301–304). International World Wide Web Conferences Steering Committee.

Yus, R., Mena, E., Ilarri, S., & Illarramendi, A. (2014a). SHERLOCK: Semantic management of location-based services in wireless environments. *Pervasive and Mobile Computing, 15*, 87–99.

Yus, R., Mena, E., Ilarri, S., Illarramendi, A., & Bernad, J. (2014b). *Multi-CAMBA: A system for selecting camera views in live broadcasting of sport events using a dynamic 3D model*. Multimedia Tools and Applications, 32 pages. Published online: 15 December 2013.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338–353.