ONTOLOGY BASED QUALITATIVE INFORMATION COLLECTION

A Thesis Submitted to
The Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Ezgi KAYSI

APRIL 2014
İZMİR

We approve the thesis of **Ezgi KAYSI**

Examining Committee Members:

Prof.Dr.Sıtkı AYTAÇ
Department of Computer Engineering
Izmir Institute of Technology

Assist. Prof Dr. Selma TEKİR
Committee Member

Assist. Prof Dr. Semih UTKU
Committee Member

27 May 2014

Prof.Dr.Sıtkı Aytaç
Supervisor, Department of Computer
Engineering
Izmir Institute of Technology

Prof. Dr. İ. Sıtkı AYTAÇ
Head of Department of Computer
Engineering

Prof. Dr. R. Tuğrul SENGER
Dean of Graduate School of
Engineering and Science

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Yrd.Doç.Dr.Tuğkan TUĞLULAR for sharing his knowledge with me and guiding me throughout my thesis. Without his thesis proposal, support, encouragement, guidance and persistence this thesis would never have happened.

Secondly, I should also express my appreciation to my manager Kurtuluş PESEN from Tüpraş for giving me this opportunity, his support and encouragement to do my thesis.

I would like to thank my close friend Gökhan KESLER for his love and great support in all steps of this thesis and for showing interest in and patience with my study.

Finally, I would like to thank my parents Metin and Havva KAYSI and my sister Beste for their unlimited patience, support and love during the this thesis and all my life.

ABSTRACT

ONTOLOGY BASED QUALITATIVE INFORMATION COLLECTION

The World Wide Web is an information resource with essentially limitless potential. However, this potential is relatively untapped because it is tough for machines to process and integrate this information meaningfully. Recently, researchers have started to explore to combine web content with specified meaning, in order to create a Semantic Web. It is one of the most encouraging and rising areas of computer science. Its capabilities to make information that is understandable to be understood and processed by machines and humans. Semantic Web approach will strictly change the effectiveness of the Internet and will enable the reuse of information and increase the representative power of information. It will be possible to combine information from different locations and process them together since they are defined in a standard way.

The potential of the Semantic Web is demonstrated with “Semantic Hotel Search” application. Also basic concepts referring knowledge with a Semantic Web language, ontology processing, reasoning and querying on ontologies are presented. Semantic Hotel Search application that contains a dynamic knowledge base holding hotel related information, which is updated at every day from the information provided in the hotels’ websites. All the information and application logic have been moved into an OWL file which controls all the content and the structure of the application. In addition to, sentimental analysis is associated with this application. Therefore, users can see what the other people said about hotels and whether user has positive or negative comment in order to make better decision.

Both travelers and travel service providers can benefit from the implemented semantic web based application. Semantic Web based travel portal can reduce search costs and a case study shows that the system provides time efficiency and is helpful in decision making.

Keywords: Semantic Web, Ontology, Sparql, Information Extraction, Semantic Hotel Domain, Sentimental Analysis, Web Services

ÖZET

ONTOLOJİ TEMELLİ NİTEL BİLGİ TOPLAMA

İnternet sonsuz potansiyeliyle bir bilgi kaynağıdır. Ancak bu potansiyel büyük oranda kullanılmamaktır çünkü makinelerin bilgileri anlamlı bir şekilde işleyip bütünleştirmesi oldukça zordur. Son zamanlarda araştırmacılar, Anlamsal Web yaratmak için web ile beli tirlen anlamını birleştirmeye başladılar. Anlamsal Web bilgisayar bilimlerinde geleceği parlak ve gelişen bir alandır. Anlamsal webin yeteneği bilgileri insanlar ve makineler tarafından anlaşabilir ve işlenebilir yapmaktır. Anlamsal Ağ yaklaşımı İnternetin etkinliğini büyük oranda arttıracak, bilginin tekrar kullanımını sağlayacak, ve bilginin sunum gücünü arttıracak. Bilgiler bir standart ile tanımlandığından, farklı yerlerdeki bilgilerin birleştirilmesi ve bu bilgilerin birlikte işlenmesi mümkün olacaktır.

Anlamsal webin gücü “Semantik Otel Arama” uygulaması ile gösterildi. Ayrıca bilginin Anlamsal Ağ dili ile sunulması, ontoloji işleme, ontolojiler üzerinden çıkarım yapma ve sorgulama kavramları gerçekleştirilmiştir. Anlamsal Otel Arama uygulaması otel ile ilgili bilgilerinin, her gün bilgilerin sağlandığı siteden güncellenerek, dinamik olarak tutmaktadır. Tüm bilgi ve uygulama mantığı uygulamanın içeriğini ve yapısını kontrol eden OWL (Web Ontoloji Dili) ontoloji dosyasına yüklenmiştir, ve bu dosya var olan mantıksal ifadelerden yeni bilgiler elde etmek için çıkarım yapmaya olanak sağlar. Bununla birlikte anlamsal otel arama uygulamasına duygu analizi de eklenmiştir. Bunun sonucunda kullanıcılar otele ait diğer insanların yaptığı yorumları görüp, aradığı kriterlerle ilgili olarak yapılan yorumların olumlu mu olumsuz mu olduğuna bakıp daha iyi seçim yapabilmektedir.

Hem seyahat edenler hem seyahat firmaları bu uygulamadan yarar sağlayacaktır. Ayrıca anlamsal web tabanlı seyahat uygulaması arama maliyetini düşürmektedir ve durum analizi göstermektedir ki bu sistem zamandan kazanç sağlamakta ve karar vermekte yardımcı olmaktadır.

Anahtar kelimeler: Anlamsal Ağ, Ontoloji, Sparql, Bilgi Çıkarımı, Anlamsal Otel Bilgi Alanı, Duygu Analizi, Web Servisleri

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

After the creation of the World Wide Web (known as WWW), presented by Tim Berners-Lee in 1989, the web was thought as some exchange of documents and data and some kind of working collaboration. It was meant to be a big working place where the programs and databases could share their knowledge and work together. But the explosion of personal computers and major advances in the field of telecommunications were the triggers of the Web as we know it today [1]. Nowadays the Web contains documents and multimedia resources concerning almost every imaginable subject, and all of this data is instantaneously available to anyone via Internet connection and the volume of information available through the web has been increasing continuously.

Its main problem is that appeared in the WWW is that the information is written only for human consumption in most of the cases. The machines cannot understand what the meaning of what is online is. A lot of pictures, drawings, movies and natural language populate the actual web. This information is not useful at all for the machines, which cannot operate with this data; they only show it to the user using a proper format.

Furthermore, users also want to use the Web to perform some task. For example, a user might want to find the best price on a desktop computer, plan and book a romantic vacation to Ibiza Island, or make reservations at a moderately-priced Chinese restaurant within three blocks of the movie they plan to see that evening. Completing these tasks often involves visiting a series of pages, integrating their content and reasoning about them in some way. This is far beyond the capabilities of contemporary directories and search engines, but could they eventually perform these tasks?

The main obstacle is the fact that the web was not designed to be processed by machines. Machines cannot perform tasks which require combining and processing data from different sources. This is why semantic web is needed.

Semantic web is an initiative by World Wide Web Consortium (W3C) in this direction to be able to make machines process such tasks. Semantic web emphasizes integration and combination of data from different data sources [2]. While the current web focuses on documents, semantic web extends the principles of web from documents to data. In this thesis, representing relevant information with a semantic web language, ontology processing, reasoning and querying on ontologies have been implemented to realize a semantic web application.

The main goal of this thesis is to extract information from the current Web which extracts relevant information from an unstructured set of HTML pages about the hotel‘s context. This information is processed in order to provide meaning to it; so the system can “understand” the texts, extract information from them, relate it and storage it. So the user can make advanced queries based on the meaning instead of the semantics. All this process of providing meaning to the unstructured texts is guided by an Ontology.

The main part of the thesis focused on Semantic Hotel Search application to show semantic web abilities and power. The main goals in Semantic Hotel Search are the integration of heterogeneous data sources and the (as far as possible) efficient and high-performance application of Semantic Web technologies in the tourism domain

The outcome of this thesis is a semantic hotel search system prototype, which uses an ontology to enrich the user search criteria. Users can search hotels based on their criteria such as review, hotel stars, location, view etc… Then more information about hotel are reachable based on sentiment analysis.

The remainder of thesis is organized as follows:

Chapter 2 gives the background information about the concept of Web, general problems and Information Management on the current Web. Then the history of Semantic Web, the domain of the Semantic Web and Semantic Web Tools and Languages are presented.

Chapter 3 is about what ontology is, Ontology Editors and Query Languages.

Chapter 4 includes System Design of Semantic Hotel Search application, System Domain and Specifications has been explained.

Chapter 5 presents the implemented ontology with Protege. Extracting information from Web via html parser. Also, explaining semantic Web services in order to update hotel information and sentimental analysis over hotel reviews.

Chapter 6 presents conclusions from the results obtained in this thesis.

CHAPTER 2

SEMANTIC WEB

2.1. Overview of the Web

The Internet has changed the style of communication with each other and the style of business is conducted. It lies at the heart of a revolution that is currently transforming the developed world toward a knowledge economy and, more broadly speaking, to a knowledge society. This development has also changed how we think about computers. Originally they were used for computing numerical calculations. Currently their predominant use is for information processing, typical applications being database systems, and games [3]. At present there is a transition of focus toward the view of computers as entry points to the information highways.

Nowadays web content is mostly suitable for human consumption. Even Web content that is generated automatically from databases is usually presented without the original structural information found in databases [4]. Typical uses of the Web today involve people's seeking and making use of information, searching for and getting in touch with other people, reviewing catalogs of online stores and ordering products by filling out forms, and viewing adult material.

These activities are not particularly well supported by software tools. Apart from the existence of links that establish connections between documents, the main valuable, indeed indispensable, tools are search engines.

Keyword-based search engines such as Yahoo and Google are the main tools for using today's Web. It is clear that the Web would not have become the huge success it is, were it not for search engines. However, there are serious problems associated with their use [5]:

- High recall, low precision. Even if the main relevant pages are retrieved, they are of little use if another 28,758 mildly relevant or irrelevant documents are also retrieved. Too much can easily become as bad as too little.
- Low or no recall. Often it happens that we don't get any relevant answer for our request, or that important and relevant pages are not retrieved. Although low recall is a less frequent problem with current search engines, it does occur.

- Results are highly sensitive to vocabulary. Often our initial keywords do not get the results we want; in these cases the relevant documents use different terminology from the original query. This is unsatisfactory because semantically similar queries should return similar results.
- Results are single Web pages. If we need information that is spread over various documents, we must initiate several queries to collect the relevant documents, and then we must manually extract the partial information and put it together.

Interestingly, despite improvements in search engine technology, the difficulties remain essentially the same. It seems that the amount of Web content outpaces technological progress [3].

But even if a search is successful, it is the person who must browse selected documents to extract the information he is looking for. That is, there is not much support for retrieving the information, a very time-consuming activity. Therefore, the term information retrieval, used in association with search engines, is somewhat misleading; location finder might be a more appropriate term. Also, results of Web searches are not readily accessible by other software tools; search engines are often isolated applications.

The main obstacle to providing better support to Web users is that, at present, the meaning of Web content is not machine-accessible. Of course, there are tools that can retrieve texts, split them into parts, check the spelling, and count their words. But when it comes to interpreting sentences and extracting useful information for users, the capabilities of current software are still very limited. It is simply difficult to distinguish the meaning of

"I am a professor of computer science."

from

"I am a professor of computer science, you may think."

2.1.1. Web Languages

A big amount of languages used to publish data in the current Web. Some of these languages are: HTML, PHP, JSP and ASP and some Media-oriented Web languages such as Flash. However, these scripting and markup languages are only meant to process the business logic of the applications and the visual presentation of the information they are dealing with. Markup languages such as HTML does not care about what the information is, it will only control the layout and appearance of the given information. Server side web scripting languages such as PHP are generally targeted to the dynamic behavior of the web applications and the business logic of such applications. However, the above mentioned languages all have a common lack in providing and processing semantic meaning bound to information. They just treat data as plain text without any meaning, that is, such web languages are not “aware” of the information they are dealing with.

2.1.2. Information Management on the Web

The incredible growth of the web has as a direct consequence of a big explosion of all kinds of online web documents. The information storage and collection on the web is as follows; the information is generally stored in large databases that are kept in the servers. The programs running on the servers generate the requested Web documents “on the fly”, based on the needed data at some state. Most of these dynamically generated on-line documents are only made for human consumption and it is impossible for the machines to understand the meaning of these documents. Such kind of Web documents is difficult to reuse and to make available to other parties because they not permanent but are being generated on specific requests without any well-defined meaning.

Most of these on-line documents are only produced for human consumption, being impossible for the machines to understand the meaning of these documents. Also manual searching by people is often a hard task and has several limitations, as it is explained on the next page, Figure 2.1.

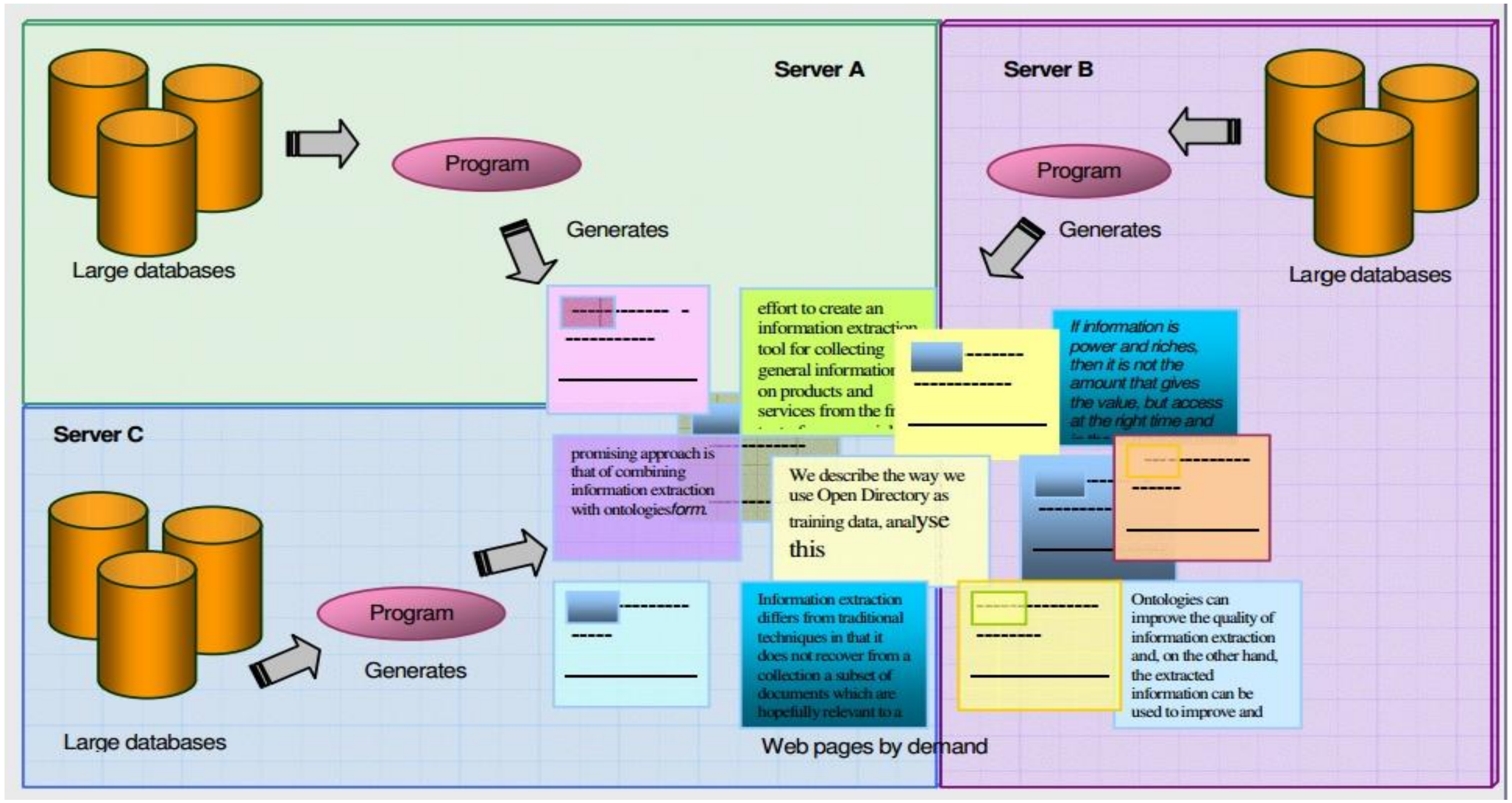


Figure 2.1 Manual Information Searching by People [1]

[1] Villarías L. G., "*Ontology-based semantic querying of the WEB with respect to food recipes*" (MS Thesis., Technical University of Denmark, 2004), 118–12

2.1.3. Information Retrieval on the Web

Human time and effort are required to browse the retrieved documents for relevant information. Current intelligent agents are unable to carry out this task in a satisfactory fashion. Information retrieval on the Web refers to the act of recovering information from the vast amount of online Web documents; getting the desired documents and presenting them to the user. This is the classic and the most widely way of obtaining information from the Web. With this approach a user does not extract any information from a document. However, the user just picks up some documents among all the available documents in the Web. The user will get a document or a set of documents and will have to analyze the document to find the desired information if it exists. Actually in this approach, only a portion of the computational power exposed by the computers is used to fetch the desired information. The computing systems used are only responsible in transferring the document and presenting to the user. No processing power is used to retrieve directly relevant information through context-aware processes and methods [3].

The problems associated with the retrieval of quality information from the Internet are many. We can consider the Internet as a connected undirected graph with many nodes where the connections are the edges. In this perspective, the nodes are distributed across the world without regard for cultures of time zones. From this point of view, the idea of a connected undirected graph captures elegantly the idea of the Internet. The problems related to traversal of the Internet to retrieve information are that the data is distributed to the whole world and the nodes of the Internet are spread across the world. And, it is obvious that the Internet is changing very fast and the data is volatile. Every six months the Internet nodes and connections are doubling in a topology that is not predefined. The data is redundant and stored in an unstructured way, and the data on the Web is duplicated in many instances across mirror sites. And also, the quality of the data is poor. The volume of data to be searched and found on the Web is growing at an exponential rate. Not all the data is in the same language because the Web is a reflection of the real world in that it is multicultural and multilingual. New media types are appearing at a fast rate, particularly where audio-visual or multimedia files are concerned. Many Web pages content's are created dynamically on demand.

On the Web, the unstructured markup languages make it difficult for humans and even more difficult for the machines to locate and acquire the desired information.

To retrieve information on the Web, current methods are browsing and keyword based searching. Both of the mentioned methods have several limitations when retrieving information from the Web.

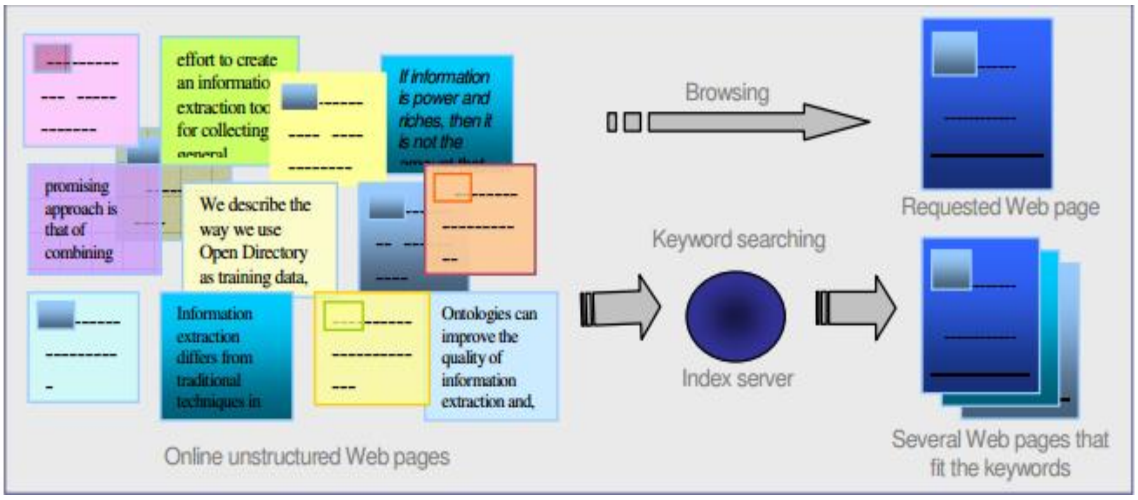

Browsing
Requested Web page
Keyword searching
Index server
Several Web pages that fit the keywords
Online unstructured Web pages

Figure 2.2 Current Web Information Retrieval[2]

Browsing: Browsing the web refers to the act of retrieving a web document by means of its URI (Uniform Resource Identifier) and displaying it in the local client browser to view its content. The user often has to traverse from link to other link in order to reach the desired information if it ever happens.

Anybody familiar with the Web knows the drawbacks of looking for information by means of browsing:

- It is very time consuming.
- It is not always possible to reach the desired information even though it exists somewhere on the Web.
- It is also very easy to get lost and disoriented following all the links the user might find relevant; suffering from what is called the “lost-in-hyperspace” syndrome.

Keyword Searching: Keyword searching is an easier way to retrieve information when compared against browsing Web documents through Web links. Keyword searching on the Web refers to the act of looking for information using some words to guide the searching. These keywords that the user wants to search for are entered into a

[2] Villarías L. G., 2004, 118–13

search engine which will perform the searching on the Web cache it has stored and indexed locally. Beforehand, the search engines continually traverse all the links available on the Web caching and indexing all the Web documents they reach. The search engines search the reduced copy of the Web following the links and trying to match the input words with the words found in its index tables. When a match occurs, the links pointed to by the index tables are returned back to the user.

Keyword searching is more useful than just browsing when looking for information, since the user does not need to know the exact URI of the desired Web document, however this approach still has some disadvantages:

- The user must be aware of the available search engines and choose the correct one that fits his/her necessities.
- The keywords entered by a user are the ones the user considers more relevant for the information he/she wants to look for, which is a very subjective decision.
- The entered keywords have to exactly match the words presented in the Web documents. Even a slight variation is not tolerated.
- Keyword searching normally returns vast amounts of useless document references/links the user has to filter by hand.

Although search engines index much of the Web's content, they have little ability to select the pages that a user really wants or needs.

2.2. Overview of the Semantic Web

The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. A web of data that can be processed directly and indirectly by machines [6].

A definition for the Semantic Web begins with defining semantic. Semantic means *meaning*. Meaning enables a more effective use of the underlying data. Meaning is often absent from most information sources, requiring users or complex programming instructions to supply it. For example, Web pages are filled with information and associated tags. Most of the tags represent formatting instructions, such as <H1> to indicate a major heading. Semantically, we know that words surrounded by <H1> tags are more important to the reader than other text because of the meaning of H1. Some Web pages add basic semantics for search engines using the <META> tag; however, they are merely isolated keywords and lack linkages to provide a more meaningful context. These semantics are weak and limit searches to exact matches. Similarly, databases contain data and limited semantic hints, if well-named tables and columns surround the data. Semantics give a keyword symbol useful meaning through the establishment of relationships.

The current Web is made up of many Web documents (pages). Any given Web document, in its current form (HTML tags and natural text), only gives the machine instructions about how to present information in a browser for human eyes. Therefore, machines have no idea about the meaning of the document they are presenting; in fact, every single document on the Web looks exactly the same to machines. Machines have no way to understand the documents and cannot make any intelligent decisions about these documents. Developers cannot process the documents on a global scale (and search engines will never deliver satisfactory performance). One possible solution is to modify the Web documents, and one such modification is to add some extra data to these documents; the purpose of this extra information is to enable the computers to understand the meaning of these documents. Assuming that this modification is feasible, we can then construct tools and agents running on this new Web to process the document on a global scale; and this new Web is now called the Semantic Web.

The Semantic Web is not Artificial Intelligence: The concept of machine-understandable documents does not imply some magical artificial intelligence which allows machines to comprehend human words and fully understand them as human beings do. Semantic Web only denotes a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to deduce people's language, it involves asking people to make the extra effort so that the machines are able to process the data in some specific way [7].

Even though it is simple to define information with languages such as RDF, at the level with the power of a Semantic Web these language will be complete languages, capable of expressing paradoxes and tautologies. And it will be possible to phrase questions whose answers normally would require a machine to search the entire Web and take an unpredictable amount of time to find the answer. Each mechanical application relying on such languages will use a schema to restrict its use to an intentionally limited language. However, when links are made between the Webs relying on such languages, the result will be an expression of a big amount of information. It is obvious that because the Semantic Web must be able to include all kinds of data to represent the world, the languages must be completely expressive.

A Semantic Web will not require every application to use expressions of arbitrary complexity [8]: Even though the languages used to define information allow expressions of arbitrary complexity and computability, applications which generate semantically defined information will in practice be limited to generating simple expressions such as access control lists, privacy preferences, and search criteria.

A Semantic Web will not require proof generation to be useful: proof validation will be enough: Although access control on Web sites involve validation of a previously prepared proof, there is no requirement for them to answer an arbitrary question, find the path and the construct of a valid proof. It is well known that to search for an answer for an arbitrary question and generate a proof for the question is typically an intractable process as many other real world problems, and a Semantic Web language does not require this (unsolvable) problem to be solved in order to be useful.

A Semantic Web is not an exact rerun of a previous failed experiment [9]: Until now other arguments has been raised against the Semantic Web concept because of the similarity to Knowledge Representation Systems. More or less, such systems have tried to achieve similar results as the Semantic Web concept is trying to do. Systems such as KIF [3](Knowledge Interchange Format) and CYC [4]are some examples of such Knowledge Representation Systems. However the success or failure of such systems should not be a threshold or limit for the Semantic Web concept/project. A more constructive approach would be to feed the Semantic Web with design experience and the Semantic Web may provide a source of data for reasoning engines developed in similar projects such as those that utilize Knowledge Representation systems.

According to above explanation, The Semantic Web is envisioned as an extension of the current web where, in addition to being human-readable using WWW browsers, documents are annotated with meta-information. This meta-information defines what the information is about in a machine processable way. The explicit representation of meta-information, accompanied by domain theories, will enable a web that provides a qualitatively new level of service. It will weave together an incredibly

[3] W3[Internet]. RDF Notes; http://www.w3.org/DesignIssues/RDFnot.html (accessed date: 21 Dec 2013)

[4] CYC [Internet]. Cycorp Home Page; http://www.cyc.com/tech.html# (accessed date : 23 Nov 2013)

large network of human knowledge and will complement it with machine processability.

FEATURE	WWW	SEMANTIC WEB
Fundamental component	Unstructured content	Formal statements
Primary audience	Humans	Application
Links	Indicate location	Indicate location and meaning
Primary vocabulary	Formatting instructions	Semantic and logic
Logic	Informal/nonstandard	Description logic

Table 2.1 Comparison of World Wide Web and Semantic Web

The flexibility and many types of Semantic Web statements allow the definition and organization of information to form rich expressions, simplify integration and sharing, enable inference, and allow meaningful information extractions *while* the information remains distributed, dynamic, and diverse. Simply put, the semantic Web improves your application’s ability to effectively utilize large amounts of diverse information on the scale of the WWW. This is accomplished through a structured, standardized approach for describing information so as to allow rich information operations.

Semantic relationships form the Semantic Web. The relationships include definitions, associations, aggregations, and restrictions. Figure 2.3 shows a small graph of statements.

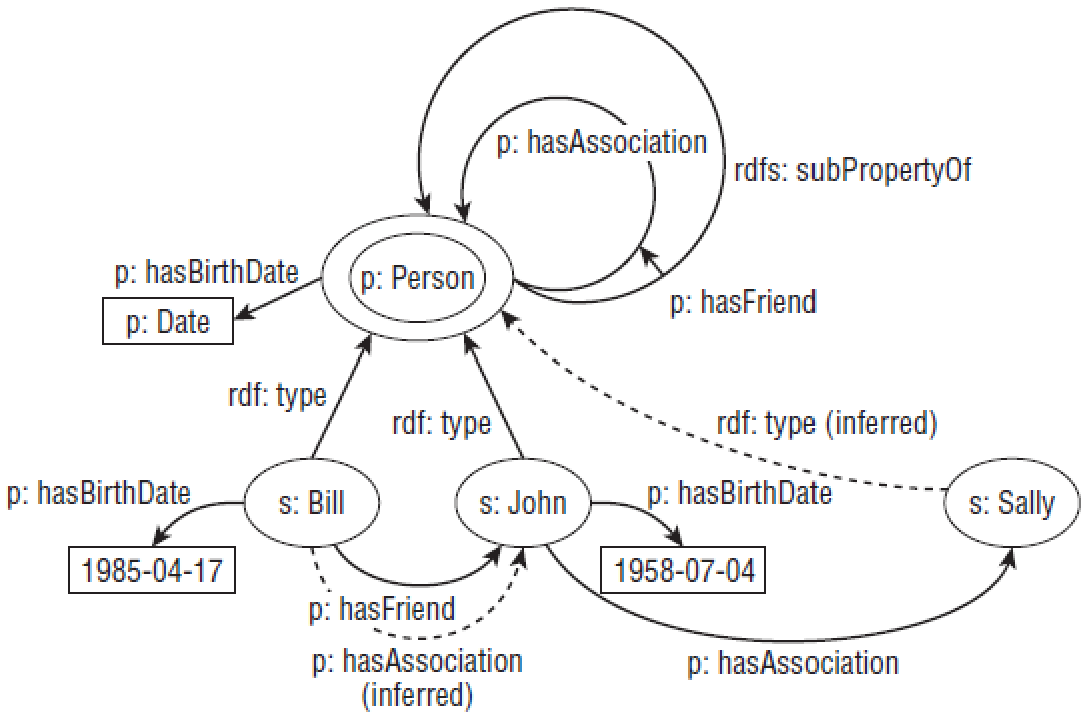


Figure 2.3 Example Graph of Semantic Relationships

A set of statements that contribute to the Semantic Web exists primarily in two forms; knowledgebases and files. Knowledgebases offer dynamic, extensible storage similar to relational databases. Files typically contain static statements. Table 2.2 compares relational databases and knowledgebases.

FEATURE	RELATIONAL DATABASE	KNOWLEDGEBASE
Structure	Schema	Ontology statements
Data	Rows	Instance statements
Administration language	DDL	Ontology statements
Query language	SQL	SPARQL
Relationships	Foreign keys	Multidimensional
Logic	External of database/triggers	Formal logic statements
Uniqueness	Key for table	URI

Table 2.2 Comparison Relational Databases and Knowledgebases

Relational databases depend on a schema for structure. A knowledgebase depends on ontology statements to establish structure. Relational databases are limited to one kind of relationship, the foreign key. The Semantic Web offers multidimensional relationships such as inheritance, part of, associated with, and many other types, including logical relationships and constraints. An important note is that the language used to form structure and the instances themselves is the same language in knowledgebases but quite different in relational databases. Relational databases offer a different language, Data Description Language (DDL), to establish the creation of the schema. In relational databases, adding a table or column is very different from adding a row. Knowledgebases really are unique because the regular statements define the structure or schema of the knowledgebase as well as individuals or instances.

2.3. Information Retrieval with Semantic Web

A lot of information is already available on the Web. We cannot expect that the entire Web will be rewritten in a structured way. This maybe is never going to happen, as the Web is not a controlled organization were some rules can be applied. Contrary it is a very decentralized and unconstrained place where everybody can post anything they want. So, there is a need new methods to gather all the spread documents and present sensible information to the user.

The role of semantics in IE is often reduced to very shallow semantic labeling. Today, most of the IE systems that involve semantic analysis exploit the simplest part of the whole view of domain and task knowledge that is to called, named entities. However, the growing need for IE application to domains that require more text understanding pushes towards more sophisticated semantic knowledge resources and thus towards ontologies viewed as conceptual models. In recent years, ontologies have emerged as a new paradigm to model and formalize domain knowledge in a machine readable way.

Ontologies are designed for being used in applications that need to process the content of information, as well as to reason about it, instead of just presenting information to humans. They permit greater machine interpretability of content than that supported by XML, RDF and RDF Schema (RDF-S), by providing additional vocabulary along with a formal semantics. So, ontologies represent an ideal knowledge background in which to base text understanding and enable the extraction of relevant information.

IE and ontologies are involved in two main and related tasks [10]:

- Ontology is used for Information Extraction: IE needs ontologies as part of the understanding process for extracting the relevant information.

- Information Extraction is used for populating and enhancing the ontology: texts are useful sources of knowledge to design and enrich ontologies.

These two tasks, as can be seen in Figure 2.4, can be combined in a cyclic process: ontologies are used for interpreting the text at the right level for IE and IE extracts new knowledge from text, to be integrated in the ontology.

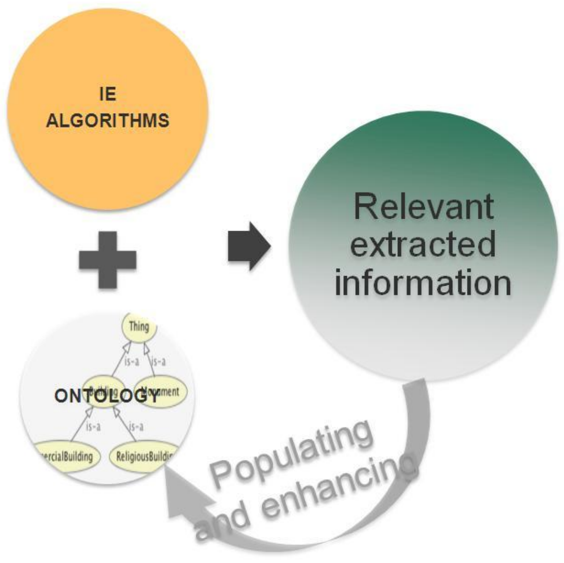


Figure 2.4 Ontology Exploitation for Information Extraction (cyclic process)[5]

This approach presents some inconveniences, but on the other hand several advantages are reached with this approach. It is very precise (very good rates of performance can be obtained when a good implementation of the ontology is made). As long as it relies on the data, if the data appearance or its order changes (and Web pages

[5] Monllao C. V., "*Ontology-based Information Extraction*" (MS Thesis., Polytechnic University of Catalonia, 2011), 104–12

usually change very often) the same application can still extract information without doing a single change.

The only dependent module is the ontology model, so if it is necessary to reconstruct the knowledge-extraction system to another subject or to another language, it is only necessary to change the ontology that describes the domain, the rest of the application will remain the same.

ADVANTAGES	DISADVANTANGES
The ontology is made manually, but only once for each domain, (it covers all web pages for the domain)	An ontology is only useful for the domain changes then the ontology has to be redefined. This has the additional work to have to make a different ontology for each topic
It is insensitive to changes in web-page format	The pages need to have some particular characteristics to apply this approach
This approach does not rely on the order or data	Another inconvenience is the language it is focus on. Ontology is a conceptual model for a certain domain in a certain language.
	Also a great knowledge of this domain is required by the ontology developer, who has to perfectly know the entities of this subject and the relations between them

Table 2.3 Advantages and Disadvantages of Semantic Web

2.4. Semantic Web Tools and Languages

The Semantic Web is built not only on mathematical theories but also on fundamental Internet technologies and philosophies. The success of the WWW has taught us not to go it alone, at least if a technology wants to survive. Build success on other proven successes. The Semantic Web is no different. The Semantic Web supports the inclusive and evolutionary nature of the WWW. The Semantic Web layer cake illustration in Figure 2.5 demonstrates some key dependencies such as URLs and XML that form the foundation of the Semantic Web.

Figure 2.5 also shows that future Semantic Web capabilities will deal with trust and providence.

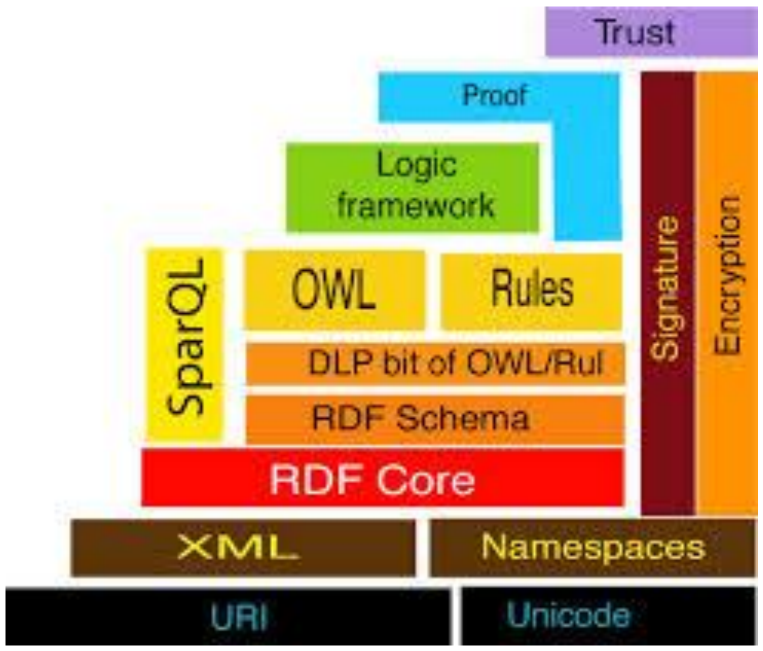


Figure 2.5 Semantic Web Layer Cake[6]

.

Figure 2.5 shows the “layer cake” of the Semantic Web (due to Tim Berners-Lee), which describes the main layers of the Semantic Web design and vision. At the bottom we find XML, a language that lets one write structured Web documents with a user-defined vocabulary. XML is particularly suitable for sending documents across the Web. RDF is a basic data model, like the entity-relationship model, for writing simple statements about Web objects (resources). The RDF data model does not rely on XML, but RDF has an XML-based syntax. Therefore, in Figure 2.5, it is located on top of the XML layer.

RDF Schema provides modeling primitives for organizing Web objects into hierarchies. Key primitives are classes and properties, subclass and sub property relationships, and domain and range restrictions. RDF Schema is based on RDF. RDF Schema can be viewed as a primitive language for writing ontologies. But there is a need for more powerful ontology languages that expand RDF Schema and allow the representations of more complex relationships between Web objects.

The Logic layer is used to enhance the ontology language further and to allow the writing of application-specific declarative knowledge.

[6] Antoniou G. Harmelen V.F., *A Semantic Web Primer* (Cambridge, MA: MIT Press, 2008), 18.

The Proof layer involves the actual deductive process as well as the representation of proofs in Web languages (from lower levels) and proof validation.

Finally, the Trust layer will emerge through the use of digital signatures and other kinds of knowledge, based on recommendations by trusted agents or on rating and certification agencies and consumer bodies. Sometimes “Web of Trust” is used to indicate that trust will be organized in the same distributed and chaotic way as the WWW itself. Being located at the top of the pyramid, trust is a high-level and crucial concept: the Web will only achieve its full potential when users have trust in its operations (security) and in the quality of information provided.

These standards are explained in detail below:

2.4.1. URI and Uniform

URI : A URI is uniform resource identifier and it is used for the identification purposes of the Web contents. Every Web content has its own URI address which is in the form of the characters or strings and it helps in recognizing the contents [11].

Uniform Resource Identifiers is simply used for identifying the resource in semantic Web URI plays an important role. The contents on the Web have a unique identity URI and it's just same as the social security number (SSN). URI consists of combination of characters and symbols.

Uniform : Uniformity delivers numerous assistances: it lets diverse kinds of resource identifiers to be used in the identical context, even also when the type of method used to access those resources may contrast and the other one is that it lets uniform semantic clarification of mutual syntactic agreements across dissimilar kinds of resource identifiers, also it lets introduction of new kinds of resource identifiers Deprived of interfering with the way that current identifiers are used; and, it lets the identifiers to be reused in numerous diverse contexts, therefore authorizing new applications or protocols to influence a big, and diversely used combination of resource identifiers.

2.4.2. XML (eXtensible Markup Language)

Despite its popularity, HTML suffered from two problems. First, whenever someone felt that HTML was insufficient for their needs, they would simply add additional tags to their documents, resulting in a number of non-standard variants [12]. Second, because HTML was mostly designed for presentation to humans, it was difficult for machines to extract content and perform automated processing on the documents. To solve these problems, the World Wide Web Consortium (W3C) developed the Extensible Markup Language (XML).

XML is already widely known, and is the basis for a rapidly growing number of software development activities. It is designed for mark-up in documents of arbitrary structure, as opposed to HTML [13], which was designed for hypertext documents with fixed structures. A well-formed XML document creates a balanced tree of nested sets of open and close tags, each of which can include several attribute-value pairs. There is no fixed tag vocabulary or set of allowable combinations, so these can be defined for each application. In XML 1.0 this is done using a document type definition (DTD) to enforce constraints on which tags to use and how they should be nested within a document. A DTD defines a grammar to specify allowable combinations and nesting of tag names, attribute names, and so on. Developments are well underway at W3C to replace DTDs with XML Schema definitions. Although XML Schema offers several advantages over DTDs, their role is essentially the same: to define a grammar for XML documents.

XML is used to serve a range of purposes:

- Serialization syntax for other mark-up languages. For example, the synchronized multimedia integration language (SMIL) is syntactically just a particular XML DTD; it defines the structure of a SMIL document. The DTD is useful because it facilitates a common understanding of the meaning of the DTD elements and the structure of the DTD.
- Separating form from content. An XML serialization can be used in a Web page with an XSL style sheet to render the different elements appropriately.
- Uniform data-exchange format. An XML serialization can also be transferred as a data object between two applications.

It is important to note that in all these applications of XML, a DTD (or an XML schema) only specifies syntactic conventions; any intended semantics are outside the realm of the XML specification.

One examples of XML text:

```
<?xml version="1.0"?>
<Gift>
<to>Peter</to>
<from>Frank</from>
<wordings>Happy Birthday!</wordings>
<body>Have a Great day</body>
</Gift>
```

At the start of the line there is declaration of the XML and its version. It is necessary to include that part in the XML code. Also we have to note down that there is no closing tag for the declaration line. So we have to keep in mind that declaration does not need closing tag.

2.4.3. RDF (Resource Description Framework)

Resource Description Framework, this is the basic framework that the rest of the Semantic Web is based on. On the Semantic Web, information is represented as a set of declarations called statements made up of three parts: subject[14], predicate, and object. Because of these three parts, statements are also sometimes referred to as triples. The fundamental concepts of RDF are resources, properties, and statements.

Resources : We can think of a resource as an object, a "thing" we want to talk about. Resources may be authors, books, publishers, places, people, hotels, rooms, search queries, and so on. Every resource has a URI, a Uniform Resource Identifier. A URI can be a URL (Uniform Resource Locator, or Web address) or some other kind of unique identifier; note that an identifier does not necessarily enable access to a resource. URI schemes have been defined not only for Web locations but also for such diverse objects as telephone numbers, ISBN numbers, and geographic locations. There has been a long discussion about the nature of URIs, even touching philosophical questions (for example, what is an appropriate unique identifier for a person?), but we will not go into detail here. In general, we assume that a URI is the identifier of a web resource.

Properties: Properties are a special kind of resources; they describe relations between resources, for example "written by", "age", "title", and so on. Properties in RDF are also identified by URIs (and in practice by URLs). This idea of using URIs to identify "things" and the relations between them is quite important. This choice gives us in one stroke a global, worldwide, unique naming scheme. The use of such a scheme greatly reduces the homonym problem that has plagued distributed data representation until now.

Statements : Statements assert the properties of resources. A statement is an object attribute-value triple, consisting of a resource, a property, and a value. Values can either be resources or literals. Literals are atomic values (strings), the structure of which we do not discuss further.

Some triple examples:

Ezgi knows Beste.
Ezgi's surname is Kaysi.
Beste knows Metin.
Gokhan works with Metin.

RDF representation will be like the figure given on the next page.

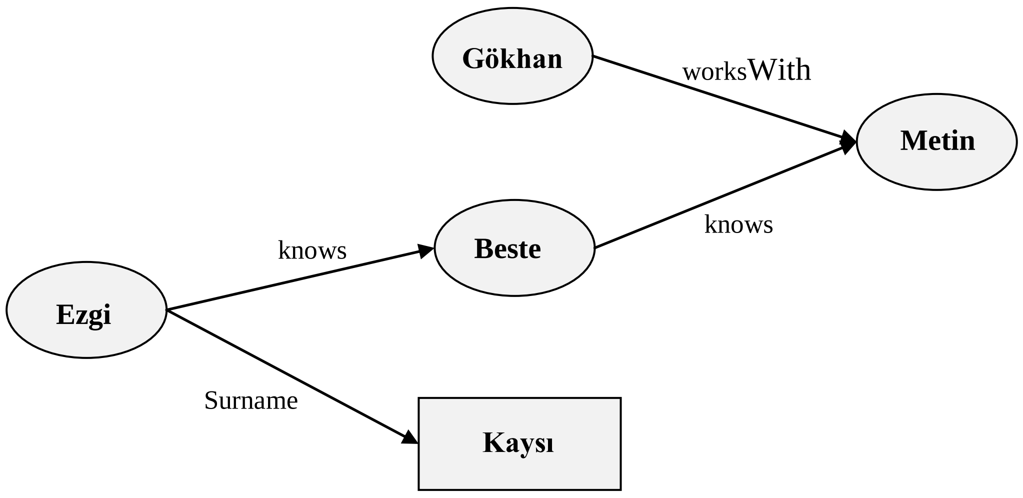


Figure 2.6 Sample RDF Representation of Triples

In the Figure 2.6 we can see that two objects are linked with each other with the predicate. In first case Gokhan is liked with Metin with a predicate “workwith” because they are colleagues while Beste is linked with Metin by predicate “knows”.

Any connection between a document and the thing(s) in the world it describes is made only by the person who reads the document. There could be a link from a document about Einstein to a document about Oslo-Norway, but there is no conception of an entity that is Einstein or linking it to the thing that is Oslo-Norway. In the Semantic Web we refer to the things in the world as resources, a resource can be anything that someone might want to talk about. Einstein, Oslo, “the value of X”, and “all the companies in Grimstad” are all examples of things someone might talk about and that can be resources in the Semantic Web. In any case, resource is the word used in the Semantic Web standards. In fact, the name of the base technology in the Semantic Web (RDF) uses this word in an essential way.

Sample Triples		
Subject	Predicate	Object
Einstein	Died	1955
Einstein	Field is	Physics
Einstein	Won	Nobel Prize

Table 2.4 Sample Triple Format

2.4.4. RDFS (RDF Schema)

RDFS is the schema language for RDF. RDF Schema extends RDF by introducing a set of distinguished resources into the language [15]. This is related to the way in which a traditional programming language can be extended by defining new language-defined keywords. But there is an important difference: XML parsers can automatically determine whether a particular XML document conforms to a given schema. Other schema languages help us to interpret particular data. For example, a database schema provides header and key information for tables in a relational database. There is neither anything in the table itself to indicate the meaning.

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-shema#">
<rdfs:Class rdf:ID="lecturer">
 <rdfs:comment>
The class of lecturers
All lecturers are academic staff members.
```

```
 <rdfs:subClassOf rdf:resource="#academicStaffMember"/>
 </rdfs:Class>
<rdf:Property rdf:ID"involves">
 <rdfs:comment>
It relates only courses to lectures
 <rdfs:comment>
 <rdfs:domain rdf:resource="#course"/>
zrdfs:range rdf: resource=#lecturer"/>
```

2.4.5. OIL (Ontology Inference Layer)

The frame structure of OIL is based on XOL (Karp et al., 1999), an XML serialization of the OKBC-lite knowledge model (Chaudhri et al., 1998). In these languages classes (concepts) are described by frames, whose main components consist of a list of superclasses and a list of slot-filler pairs. OIL extends this basic frame syntax so that it can capture the full power of an expressive description logic. These extensions include the following:

- Arbitrary Boolean combinations of classes (called class expressions) can be formed, and used anywhere that a class name can be used. In particular, class expressions can be used as slot fillers, whereas in typical frame languages slot fillers are restricted to being class (or individual) names.
- A slot-filler pair (called a slot constraint) can itself be treated as a class: it can be used anywhere that a class name can be used, and can be combined with other classes in class expressions.
- Class definitions (frames) have an (optional) additional field that specifies whether the class definition is primitive (a subsumption axiom) or nonprimitive (an equivalence axiom). If omitted, this defaults to primitive.
- Different types of slot constraint are provided, specifying value restriction, existential quantification and various kinds of cardinality constraint (some frame languages also provide this feature, referring to such slot constraints as facets).
- Global slot definitions are extended to allow the specification of super slots (subsuming slots) and of properties such as transitive and symmetrical.
- Unlike many frame languages, there is no restriction on the ordering of class and slot definitions, so classes and slots can be used before they are ‘defined’’. This means that OIL ontologies can contain cycles.

- In addition to standard class definitions (frames), OIL also provides axioms for asserting disjointness, equivalence and coverings with respect to class expressions (and not just with respect to atomic concepts).

Many of these points are standard for a DL, but are novel for a frame language. OIL is also more restrictive than typical frame languages in some respects. In particular, it does not support collection types other than sets (e.g. lists or bags), and it does not support the specification of default fillers. These restrictions are necessary in order to maintain the formal properties of the language (e.g. monotonicity) and the correspondence with description logics.

2.4.6. DAML+OIL (DARPA Agent Markup Language – OIL)

These two languages are the XML and Web-based languages to support the development of Semantic Web.

DAML+OIL is a descriptive semantic markup language for Web resources which is built on top of earlier defined languages such as RDF and RDF Schema, and extends these languages with richer modeling primitives enabling reasoning systems to process it more effectively [16]. DAML+OIL was developed by the Defense Advanced Research Projects Agency (DARPA) under the DARPA Agent Markup Language (DAML3) Program.

With DAML+OIL, in order to make the information/data yet more expressive and powerful, it is possible to use description logic to describe the data enabling it to be processed on reasoning systems. In this way, not only the explicitly given data will be available but some new facts and conclusions will be available about the data provided. In order to achieve this extra feature, the DAML+OIL is a suitable language because of its expressiveness with descriptive logic. For achieving this extra power, an extension of RDF, called DAML+OIL, can be used. DAML+OIL is a description logic language disguised in an XML format.

DAML extends RDFS in the following ways:

- Support of XML Schema data types rather than just string literals and primitive data types such as dates, integers, decimals, etc.
- Restrictions on properties like cardinality constraints.
- Definition of classes by enumerations of their instances.

- Definition of classes by terms of other classes and properties. In order to enable the definition from other classes different expressions has been defined such as; unionOf, intersectionOf, complementOf, hasClass and hasValue which some of them has their roots in classic set theory.
- It is possible to make Ontology and instance mappings (sameClassAs, samePropertyAs, sameIndividualAs, differentIndividualFrom) permitting translation between ontologies.
- Additional hints to reasoning systems such as; disjointWith, inverseOf, TransitiveProperty and the UnambiguousProperty.

DAML is not completely developed yet. Even though it was actually the recommended ontology language by the World Wide Web Consortium, a new project called Ontology Web Language (OWL) has been developed to replace DAML. The OWL project has removed some of the requirements specified for DAML language, as rules, queries and services are still under development.

Description logics (DLs) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain and is very well-suited to provide structure to information [17]. Description Logics is a subset of First Order Logic, which is non functional and does not allow explicit variables. It is less expressive in favor of having greater decidability when processed by inference procedures. Description Logics is different form predecessors, such as semantic networks and frames, in that they are equipped with a formal, logic-based semantics.

High quality Web ontologies are necessary for the Semantic Web to be successful, and their construction, integration, and evolution is greatly dependent on the availability of a well-defined semantics and powerful reasoning systems. Since DLs provide these aspects, they should be ideal candidates for creating and developing ontology languages. That much was already clear ten years ago, but at that time, there was a fundamental mismatch between the expressive power and the efficiency of reasoning that DL systems provided, and the expressivity and the large knowledge bases that ontologists needed. Through the basic research in DLs in the last 10 to 15 years, the gap between the needs of ontologists and the systems that DL researchers provide has finally become narrow enough to build stable bridges.

2.4.7. OWL (Web Ontology Language)

OWL (Web Ontology Language) is the latest recommendation of W3C [18], and is probably the most popular language for creating ontologies today. It is also the last technical component we need to familiarize ourselves with. The good news is that it is built on RDF schema; as you have already established a solid understanding of RDF schema, much of the material here is going to be looked familiar.

OWL = RDF schema + new constructs for expressiveness.

Therefore, all the classes and properties provided by RDF schema can be used when creating an OWL document.

OWL and RDF schema have the same purpose: to define classes, properties, and their relationships. However, compared to RDF schema, OWL gives us the capability to express much more complex and richer relationships. The final result is that you can construct agents or tools with greatly enhanced reasoning ability.

Therefore, we often want to use OWL for the purpose of ontology development; RDF schema is still a valid choice, but its obvious limitations compared to OWL will always make it a second choice.

OWL Lite: OWL Lite is targeted for users only needing simple constraint features and classification hierarchies. For example, even though OWL Lite supports cardinality constraints the cardinality values are restricted. For such constraints only the values 0 and 1 is allowed. It is much simpler to provide tool support for OWL Lite than it is for its more expressive relatives. This will allow easy migration to OWL Lite from different ontology languages being used on the market.

OWL DL: OWL DL supports users who want the maximum expressiveness without the lack of computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL includes all OWL language constructs with restrictions such as type separation (a class cannot also be an individual or property, a property cannot also be an individual or class) enabling to create distinct definitions. It is named as OWL DL because of its correspondence to Description Logic, a field of research that has studied a decidable fragment of first order logic. OWL DL was designed so that it has desirable computational properties for reasoning systems.

OWL Full: OWL Full is targeted for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees[19]. Decidability and completeness properties have not been restricted as it is in OWL DL. Type separation is not as strict as it is in OWL DL. For example, in OWL Full a defined class can be treated as a collection of different individuals and as an individual in its own right simultaneously. Another important difference from OWL DL is that in OWL Full a owl:DatatypeProperty can be marked as an owl:InverseFunctional-Property. OWL Full allows an ontology to incorporate the meaning of a pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature supported by OWL Full.

Each of the sublanguages mentioned above is an extension of their simpler predecessor; both in what can be legally expressed in the ontology and in what can be validly concluded in it. The following set of relations hold, but their inverses do not.

Every legal OWL Lite ontology considered legal OWL DL ontology.

Every legal OWL DL ontology considered legal OWL Full ontology.

Every valid OWL Lite conclusion considered valid OWL DL conclusion.

Every valid OWL DL conclusion considered valid OWL Full conclusion.

Ontology developers should consider which of the species best suits their needs when choosing an OWL language. When making choice between OWL Lite and OWL DL, the choice depends on whether the users need the more expressive restriction constructs provided by OWL DL. Reasoning systems for OWL Lite will have desirable computational properties. Reasoners for OWL DL will be subject to higher worst-case complexity because of its more expressiveness compared to OWL Lite. When considering OWL DL and OWL Full, the choice between them mainly depends on the extent to which users require the meta-modeling facilities provided by RDF Schema (i.e. defining classes of classes). Reasoning support is less predictable when comparing OWL Full to OWL DL.

Moreover, OWL makes an open world assumption, that is, descriptions of resources are not bounded to a single file or scope. While class C1 may be defined originally in the ontology O1, it can also be extended in other ontologies. The consequences of these additional propositions about C1 are not reversible. New information cannot retract previous information where the new information is originated from. New information from reasoning can be contradictory, but facts and entailments can only be added and never deleted.

It is the responsibility of the designer of the ontology to take into consideration, the possibility of these contradictions. It is expected that tool support will help detecting such cases. In order to write an ontology that can be interpreted unambiguously and used by software agents, a syntax and formal semantics for OWL is required. In addition, OWL is a vocabulary extension of RDF.

CHAPTER 3

ONTOLOGYAND QUERY LANGUAGES

Ontology describes the concepts in the domain and also the relationships that hold between those concepts[20]. Different ontology languages provide different facilities. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C). There are several definitions of ontology each and one differ from each other.

Ontologies are designed for the purpose of defining knowledge, reusing and sharing it effectively. It is a formal definition for some ontological commitment so that different parts can participate when relying on the same definitions and vocabularies. An ontology is a set of definitions written using a formal vocabulary [21]. To specify a conceptualism this is the main approach used because it has some properties enabling AI processing systems to share knowledge among them. In other words, an ontological commitment is a kind of an agreement involving different domain specifications to use a specific vocabulary when defining concepts. Different processing systems are being built so that they can participate in such commitments. That is they can be "connected" to some ontology without any conflicts with respect to the definitions and the vocabularies used. An ontology is built so that such systems can participate into them, and share knowledge among other systems.

Specified a significant domain, an ontology defined for that domain is the base for the knowledge to represent for that domain. An ontology enables the definition of a vocabulary in order to express the knowledge for some domain. Without a definition of some vocabulary it is not possible to share knowledge among different systems/agents. Simply said, there will be no common ground for such systems to exist and share knowledge. A domain is a specific area of a subject or an area of knowledge like travel, medicine, economy a specific research etc. Ontologies are used by systems/agents such as databases application programs or any other thing that need to share knowledge. Ontologies are being build up of basic concepts and the different relations between them. The definitions of such concepts and relation are computer usable so that computing systems can process these concepts and relations.

Another definition can be “ontology is a formal explicit description of concepts in a domain of discourse. Classes which are sometimes called concepts, properties of these classes are call slots or roles, while restrictions on slots are called facets. When ontology is together with the instances it creates knowledge base [22]. Ontology is truly based on the representation of classes. So it can be said that classes give concept to ontology.

Class definitions are the most common approach when defining some domain in an ontology. Class definitions are suitable to define and describe the different concepts within a domain. For example, a class defining a pizza represents all the different pizza instances that exist. Any pizza is an instance of the class defining and describing a pizza. Classes can have inheritance relations between them enabling the definition of more specific classes from a given class. Definition of more general classes is also possible. For example we can have subclasses of the class pizza as “spicy pizza” and “non spicy pizza” where the class pizza is a super-class of these two classes.

Since the beginning of the 1990s ontology became an admired research topic investigated by quite a few AI research communities including knowledge engineering, natural language processing, and knowledge representation. More recently, the notion of ontology is also becoming extensive in the fields like intelligent information integration, cooperative information systems, information retrieval, electronic commerce, and knowledge management [23].

In computer science and information science, ontology is a formal representation of the knowledge by a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to describe the Domain. An ontology supports software agents, or in general all computer systems requiring to share and reuse domain knowledge. Important features of an ontology are listed on below [24]:

- Ability to reuse domain language.
- Making domain assumptions explicit.
- Separation of operational knowledge and domain knowledge.
- Sharing the formal definitions and vocabularies when describing some concept.
- Analysis of domain knowledge.

Components of an Ontology

Most ontology describes individuals (instances), classes (concepts), attributes, and relations. Common components of ontologies include [25]:

- Individuals: Individuals are also known as instances. Individuals can be referred to as being `instances of classes'. Individuals, represent objects in the domain that we are interested in.

- Classes: Sets, collections, concepts, classes in programming, types of objects, or kinds of things. Classes is described using formal descriptions that state requirements for membership. It may be organized into superclass- subclass hierarchy.

Example: Dog is a subclass of Animal. Superclass- subclass relationships may be computed by a reasoner.

- Attributes: Aspects, properties, features, characteristics, or parameters that objects (and classes) can have. Another definition can be binary relations on individuals. It can be transitive or symmetric.

- Relations: Ways in which classes and individuals can be related to one another

- Function terms: Complex structures formed from certain relations that can be used in place of an individual term in a statement.

- Restrictions: Formally stated descriptions of what must be true in order for some assertion to be accepted as input.

- Rules: Statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form.

- Axioms: Assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application.

- Events: The changing of attributes or relations.

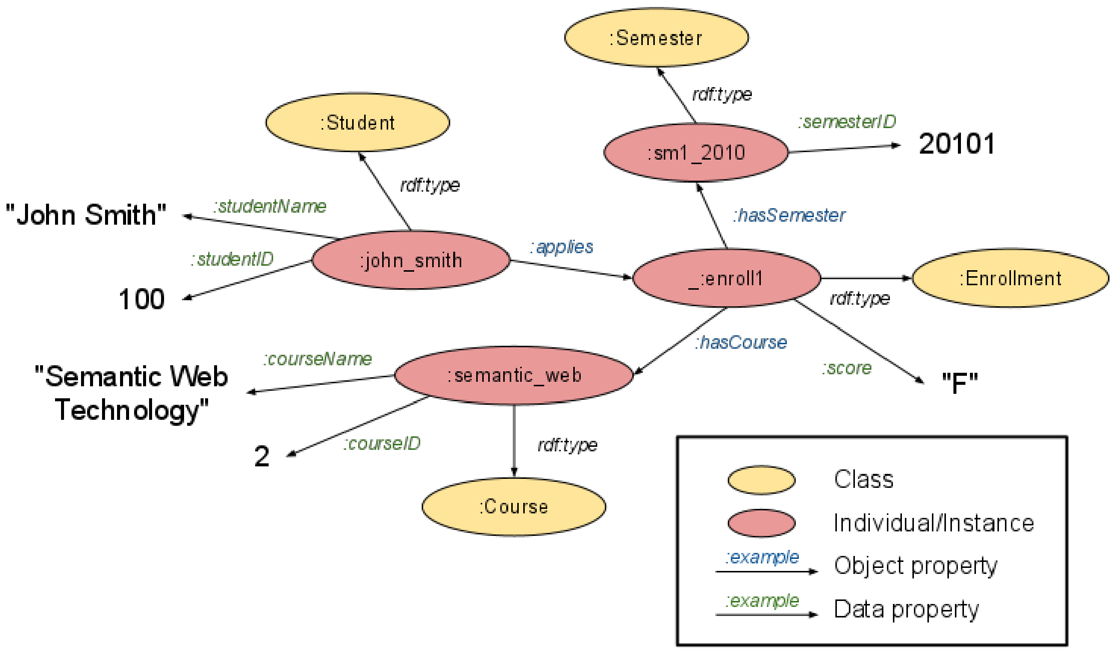


Figure 3.1 Relations among Some Classes and Instances

Some classes, instances, and relations among them in the enrollment ontology domain. Yellow color was used for classes and pink for instances. Links represent data property and object property for relation of between two classes.

Sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontologies. For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

Enabling reuse of domain knowledge was one of the driving forces behind recent surge in ontology research. For example, models for many different domains need to represent the notion of time [26]. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops such an ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain.

Making explicit domain assumptions underlying an implementation makes it possible to change these assumptions easily if our knowledge about the domain changes

[27]. Hard-coding assumptions about the world in programming-language code makes these assumptions not only hard to find and understand but also hard to change, in particular for someone without programming expertise. In addition, explicit specifications of domain knowledge are useful for new users who must learn what terms in the domain mean.

Separating the domain knowledge from the operational knowledge is another common use of ontologies. We can describe a task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves [28]. We can then develop an ontology of PC-components and characteristics and apply the algorithm to configure made-to-order PCs. We can also use the same algorithm to configure elevators if we "feed" an elevator component ontology to it.

Analyzing domain knowledge is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable when both attempting to reuse existing ontologies and extending them [29].

Often an ontology of the domain is not a goal in itself. Developing an ontology is akin to defining a set of data and their structure for other programs to use. Problem-solving methods, domain-independent applications, and software agents use ontologies and knowledge bases built from ontologies as data.

What are the differences between ontologies and relational databases?

Although databases and ontologies have some similarities, they differ in many important features. First of all an ontology is not storage for data but is a defining model for the data whereas a relational database is a data repository. An ontology can be used as filter or a framework to access and manipulate data where a database can be used to store the different data instances defined by the ontology [30]. Another important difference is querying. When making queries against a relational database the returned data will be the same data stored previously, just matching some conditions. However when making a query against an ontology, together with some reasoning process, the returned data can be some inferred data which was not stored previously but generated from some facts represented by the ontology. In ontologies, queries can also be made for some specific relations while this is not possible with ordinary relational databases.

Ontologies should be well designed and also well defined. By a good design it is meant that they should adequately capture the modeled domain, be understandable for a human user and provide good support for machine processing [31]. With a good definition it is meant not only the syntax, but also the semantics. The formal semantics is important that if we want to introduce an automated reasoning over ontologies. Such reasoning enables to support the ontology design (such as consistency checking or supporting more authors developing one ontology), integrating and sharing ontologies automatically, determining and establishing relationships among ontologies etc.

Types of Ontologies

Ontologies can be classified from at least two aspects, by the type of information they capture, and by the richness of their internal structure. The former is divided into six groups [32]:

Top level ontologies or upper-level ontologies are the most general ontologies describing the top-most level in ontologies to which all other ontologies can be connected, directly or indirectly. In theory, these ontologies are shareable as they express very basic knowledge, but this is not the case in practice, because it would require agreement on the conceptualization of being, which is very difficult. Any first step from the top-most concept would be either too narrow, thereby excluding something, or too broad, thereby postponing most of the problem to the next level, which means of course that controversial decisions are required.

Domain ontologies describe a given domain, e.g. medicine, agriculture, politics, etc. They are normally attached to top-level ontologies, if needed, and thus do not include common knowledge. Different domains can be overlapping, but they are generally only reusable in a given domain. The overlap in deferent domains can sometimes be handled by a so-called middle-layer ontology, which is used to tie one or more domain ontologies to the top-level ontology.

Task ontologies define the top-level ontologies for generic tasks and activities.

Domain-task ontologies define domain-level ontologies on domain specific tasks and activities.

Method ontologies give definitions of the relevant concepts and relations applied to specify a reasoning process so as to achieve a particular task.

Application ontologies define knowledge on the application level and are primarily designed to fulfill the need for knowledge in a specific application [33].

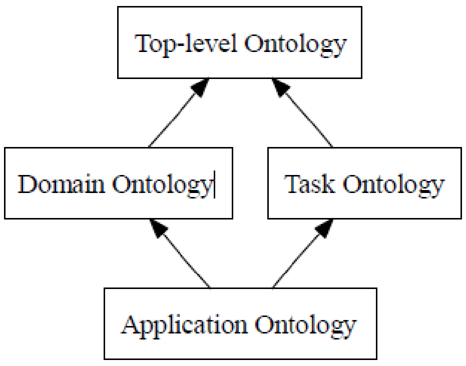


Figure 3.2 Guarino's Kinds of Ontologies[7]

3.1. Ontology Editors

Effective and efficient work with the semantic Web must be supported by advanced tools enabling the full power of this technology. In particular, it requires the following elements [34]:

- Formal languages to express and represent ontologies (We already discussed some of these in the last section)
- Editors and semiautomatic construction to build new ontologies
- Reusing and merging ontologies (ontology environments that help to create new ontologies by reusing existing ones)
- Reasoning services (instance and schema inferences that enable advanced query answering service, support ontology creation, and help map between different terminologies)
- Annotation tools to link unstructured and semi structured information sources with metadata
- Tools for information access and navigation that enable intelligent information access for human users

[7] Reciis [Internet]. *Ontology and its inference in description logics*; http://www.reciis.icict.fiocruz.br/index.php/reciis/article/viewArticle/339/759 (accessed date: 11 Dec 2013)

- Translation and integration services between different ontologies that enable multistandard data interchange and multiple view definitions (especially for B2B electronic commerce).

Ontology editors help human knowledge engineers build ontologies. They support the definition of concept hierarchies, the definition attributes for concepts, and the definition of axioms and constraints. They enable the inspection, browsing, codifying, and modification of ontologies and in this way support the ontology development and maintenance task. To be useful in this context, they must provide graphical interfaces and must conform to existing standards in Web-based software development.

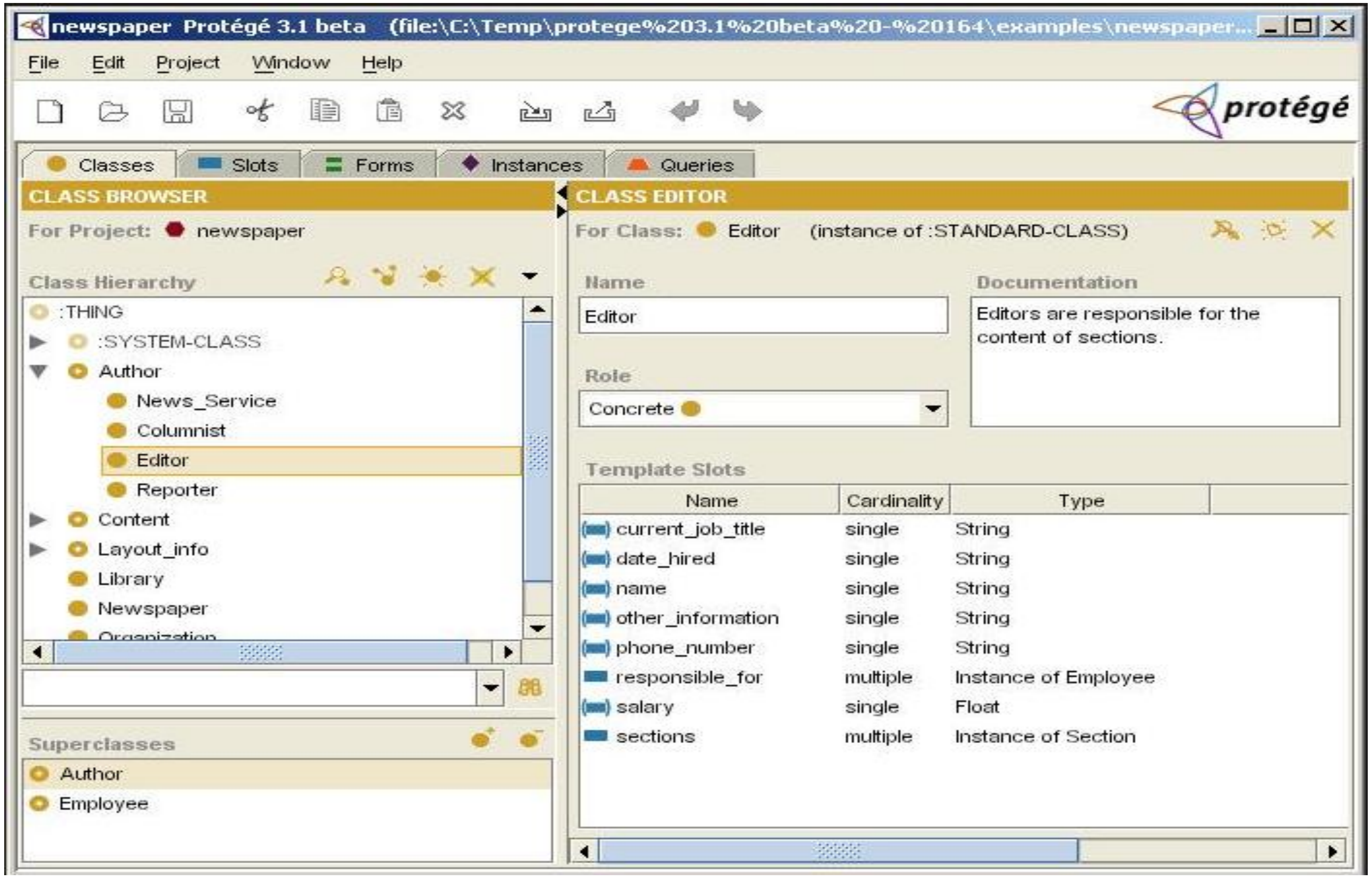
Today, there are more than 90 tools available for ontology development from both non-commercial organizations and commercial software vendors [35] [36]. Most of them are tools for designing and editing ontology files. Some of them may provide certain capabilities for analyzing, modifying, and maintaining ontologies over time, in addition to the editing capabilities. One of the more popular editing tools is Protégé, developed by the Stanford University School of Medicine [37]. Other tools are SemTalk, OilEd , Unicorn, Jena, and Snobase, to name a few.

Some of the different tools available can be integrated to each other enabling a more complete development environment. For example the ontology editor Protégé can communicate with the Inference engine to make reasoning and consistency checking on the ontology being built.

Protégé: Protégé [8]is free ontology editor which is created by Stanford University. It is very popular tool for editing and creating ontology. It supports Frame based ontology design and OWL ontology. The plus point for protégé is that it has several plug-ins which is very handy and useful. Due to its plug-ins it is used a lot all over the world. Protégé has a frame-based knowledge model, which is completely compatible with OKBC (The Open Knowledge-Base Connectivity protocol) enabling interoperability with other knowledge-representation systems. Protégé enables a development environment supported by a number of third party plug-in, targeted to the specific needs of specific knowledge domains. It is also an ontology development platform which can easily be extended to include various graphical components such as

[8] ProtégéWiki [Internet]. *WebProtégé*; http://protegewiki.stanford.edu/wiki/WebProtege (accessed date: 11 Jan 2014)

graphs and tables, media such as sound, images, and video, and various storage formats such as OWL, RDF, XML, and HTML.



Name: Editor

Documentation: Editors are responsible for the content of sections.

Name	Cardinality	Type
current_job_title	single	String
date_hired	single	String
name	single	String
other_information	single	String
phone_number	single	String
responsible_for	multiple	Instance of Employee
salary	single	Float
sections	multiple	Instance of Section

Figure 3.3 Screenshot of Protege 3.1 Beta

Web protégé: Web protégé is also a product of Stanford centre but it is used for developing web ontology. Its user interface is just as same as Protégé so it's online version of protégé.

Ontolingua: The Ontolingua system [38] [39] provides users with the ability to manage, share and reuse different ontologies stored on a remote ontology server. The system has been developed at the Knowledge Systems Laboratory at Stanford University in the early 90s. Ontolingua supports a wide range of translations while most ontology editors only support a limited range of translations. It can easily import and export constructed ontologies with the newer languages like DAML+OIL and OWL.

WebOnto: WebOnto [40] can manage ontologies constructed in OCML. It is a Web-based tool for browsing, editing and managing ontologies constructed with OCML. WebOnto has been developed at the Knowledge Media Institute, at the Open University as part of several European research projects in the late 90s. It is basically a Java based client application connected to a specific Web server having access to ontologies constructed with OCML.

NeOn Toolkit: NeOn is an ontology engineering tool. Just like Protégé it has plug-ins but which are lesser in amount as protégé. Basically this tool is used for making weighty projects for example Ontology mapping and integration, multi-modular ontologies etc.

SWOOP: It is a small but very simple tool for developing ontology. Due to its simplicity it is majorly used by beginners for developing ontology.

The tool used in this thesis is called Protégé and there are a lot of reasons for using the protégé and those are;

- **User Friendly user interface:** The interface for protégé is very simple for starters in ontology engineering. The workspace can be modified for more simplicity and rapid work.
- **Scalability:** In Protégé user can develop from small ontology to heavy weight ontology containing several thousands of classes.
- **Plug-in Architecture:** The plug-in architecture in Protégé is very good plus point for protégé which differentiate it from other tools. The plug-ins are used for several purpose. The plug-ins can be used for visualizing the ontology. For example the OWLviz and OntoGraf are the type of plug-ins which helps in visualizing the created ontology. On next pages, In Figure3.4 and Figure3.5 we can see the visualization

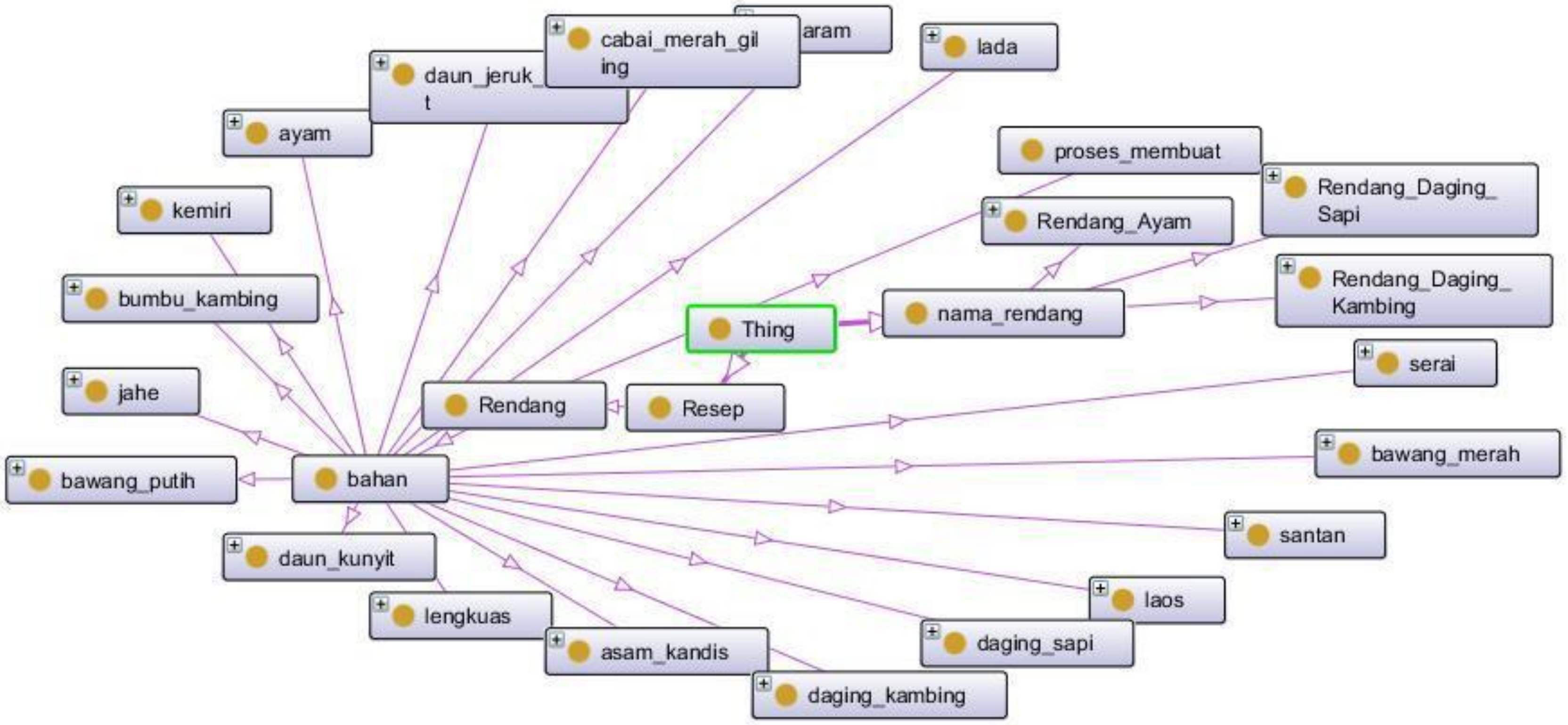


Figure 3.4 Sample Onto Graph Visualization[9]

[9] Shahzad M., "*A semantic web approach for dealing with university courses*" (MS Thesis., University of Agder, 2013), 166–24

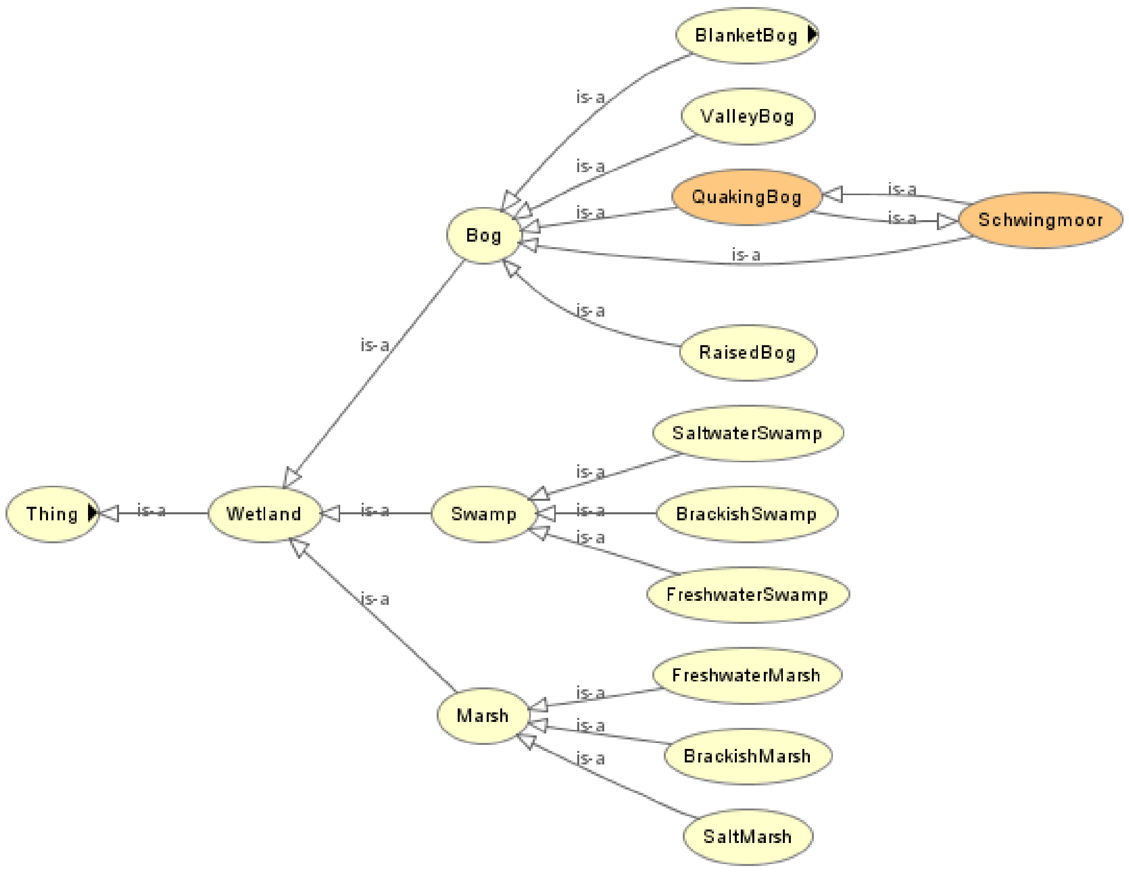


Figure 3.5 Sample OWLViz Visualization[10]

Not only visualization but there are more plug-ins which works very differently. There are some more useful plug-in listed below

- DL Query
- Ontology Differences
- OWL Code Generation

3.2. Ontology Management System

An ontology management system for ontologies is similar to a database management system for relational databases. A DBMS allows an application to access data stored in a database via a standard interface. The techniques for storing and structuring the data is left to the DBMS itself so that the application does not have to consider these issues. The DBMS system allows the application to access data stored in

[10] Shahzad M., "*A semantic web approach for dealing with university courses*" (MS Thesis., University of Agder, 2013), 166–24

the database with a query language (SQL) taking care of all the things related to data storage, indexing of data and data file management. An ontology management system allows access to ontologies in a similar way that a DBMS does. The application making queries on an ontology through an ontology management system does not have to worry about how the underlying processes relate to data storage, and how structuring of data is done. Ontology editing capabilities are not the central parts of an ontology management system, however some systems may provide capabilities to edit ontologies programmatically through a programming interface. In the case that such editing capabilities are not provided, developers can choose to use some graphical editing environments such as Protégé.

Jena Framework: Jena[11] is a toolkit for developing applications within the semantic web. Jena is implemented in the Java programming language. These Java-based abstractions translate the statements and constructs of the Semantic Web into useful programming artifacts such as Java classes, objects, methods, and attributes. Jena provides several Reasoner types to work with different types of ontologies. Some important capabilities of Jena are listed below [41]:

- Provides a RDF Application Programming Interface (API).
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- Provides an OWL API.
- Provides both in-memory and persistent storage ontology models
- Provides support for RDQL – a query language for RDF

In addition on the fine grain access provided by the Jena API, SDB can be coupled with the web-server - 'Joseki' - which is SPARQL query server. This enables an SDB store to be queried over HTTP. Jena recently introduced a non-transactional native store called TDB. For our current evaluation we have Jena SDB backed with MySQL.

Sesame: Sesame[12] is an open source framework for storage, inferencing and querying of RDF data. Sesame matches the features of Jena with the availability of a

[11] Jena Apache [Internet]. *Jena Ontology API*; http://jena.apache.org/documentation/ontology/ (accessed date: 11 Jan 2014)

[12]Bioontology[Internet].*Sesame*;http://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf (accessed date: 11 Jan 2014)

connection API, inferencing support, availability of a web server and SPARQL endpoint. Like Jena SDB it provides support for multiple back ends like MySQL and Postsgre. Sesame Native is the native triple store offering from Sesame. As compared to be Jena’s native triple, TDB, it is less scabable mentions that Sesame native has been tested with up to ~70 Million triples while Jena TDB is known to work with 1.7 Billion triples [42].

LinqToRDF: LinqToR[13]DF is a Semantic Web framework for .NET. It provides an easy way to integrate Semantic Web queries into your software. At the core of the system sits a LINQ query provider (like LINQ to SQL) that converts your queries into the SPARQL query language. You don't have to know that much SPARQL or RDF to be able to use it. It also provides a UML-style design surface allowing you to create RDF files, and to generate compatible C# code to work with the RDF.

It provides developers with an intuitive way to make queries on semantic web databases. The project has been going for a year and it's starting to be noticed by semantic web early adopters in the .NET community. LINQ provides a standardized query language and platform enabling any developer to understand systems using semantic web technologies via LinqToRDF. It will help those who don't have the time to ascend the semantic web learning curve to become productive quickly.

FRAMEWORK	LANGUAGE/INTERFACE	SEMANTIC LEVEL
Jena	Java	RDF to OWL
Sesame	Java & RESTful web service	RDF
OWL API	Java	OWL 2
RAP-RDF API	PHP	RDF
Redland	C,Python,Ruby,Perl, & PHP	RDF
LinqToRDF	.NET	RDF

Table 3.1 Semantic Web Framework Summary

[13] Google [Internet]. *linqtordf*; https://code.google.com/p/linqtordf/ (accessed date: 11 Jan 2014)

3.3. Ontology Query Languages

Based on the ontology languages described above, several query languages and systems have been already developed. RDQL is the query language for Jena2. Vampire is a FOL theorem prover using TPTP3 format for problem input and query input. nRQL is the query language for RACER. Finally, OWL-QL is a query language for the OWL-QL system. SPARQL is a Server-Client-based RDF query language. It has SQL syntax and is influenced by RDQL and SquishQL4 . SPARQL supports disjunction in the query and thus can process more complex query than RDQL. SPARQL also provides optional variable binding and result size control mechanisms for real world usage. However, we did not choose SPARQL as a query language for testing, since there was no supporting system available at the time [43].

OWL-QL (OWL Query Language): OWL-QL is a query language and protocol supporting agent-to-agent query-answering dialogues using knowledge represented in OWL language [44]. The semantic relationship is exactly specified among a query, a query answer and the included ontologies to produce query answer. It also provides dialog with the query engine so that the query engine can use automated reasoning methods to derive answers to queries. The query engine could need some extra information from the querying agent in order to produce the answer. So a dialog could exist between the two parts in order to produce the answer to a query. This is why OWL-QL has the properties as a protocol. In this setting, the set of answers to a query may be of unpredictable size and may require an unpredictable amount of time to compute since the domain is not totally restricted because multiple knowledge bases can be involved in the dialog between query agents. The following quote is from the OWL-QL specification; “an OWL-QL query contains a query pattern that is a collection of OWL sentences in which some literals and/or URI-refs have been replaced by variables. A query answer provides bindings of terms to some of these variables such that the conjunction of the answer sentences – produced by applying the bindings to the query pattern and considering the remaining variables in the query pattern to be existentially quantified – is entailed by a knowledge base (KB) called the answer KB”.

OWL-QL is relatively simple and expressive. To make a query, a querying agent can simply describe what is being searched for, indicating the variables and their matching concepts for the answer queried. The advantage of the OWL-QL query language is that the underlying mechanism is easily adaptable for different ontology representation languages.

RDQL and Jena2: RDQL is a query language for RDF in the Jena framework. The development of RDQL is to provide a data-oriented query model. This means that RDQL only retrieves information stored in the model which contains a set of N-Triple statements. RDQL provides no reasoning mechanisms. The reasoning is provided by user selected reasoners bound to the model containing the original ontology information. Provided with a proper reasoner, RDQL can process ontology in various languages including OWL. SPARQL RDQL predates SPARQL - in fact, RDQL design predates the current RDF specifications and some of the design decisions in RDQL are a reflection of that. The biggest of these is that RDF didn't have any data typing so RDQL handles tests on, say, integers without checking the data type (if it looks like an integer, it can be tested as integer). SPARQL has all the features of RDQL and more [45]:

- ability to add optional information to query results
- disjunction of graph patterns
- more expression testing (date-time support, for example)
- named graphs
- sorting

However, above all, it is more tightly specified so queries in one implementation should behave the same in all other implementations.

RDF is a directed, labeled graph data format for representing information in the Web. This specification defines the syntax and semantics of the SPARQL query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports aggregation, sub queries, negation, creating values by expressions, extensible value testing, and constraining queries by source RDF graph. The results of SPARQL queries can be result sets or RDF graphs.

```
# George Washington’s Namesakes
SELECT ?location
WHERE {
?person <http://www.w3.org/2000/01/rdf-schema#label>
"George Washington"@en.
?location <http://dbpedia.org/property/namedFor> ?person
     }
```

CHAPTER 4

ONTOLOGY BASED QUALITATIVE INFORMATION COLLECTION

In the previous chapter we introduced the vision of the Semantic Web. This chapter will go through the design and specifications of the Semantic Web application project implemented for this thesis. Specifications regarding choice of domain, services, user facilities etc. will be discussed in detail in this chapter.

As the Internet has been in a period of rapid growth, the need of applications making use of machine and human consumable data has come on to the scene as a promising candidate for a new breakthrough in information presentation and processing. Various kind of technologies have been developed along with different standards and techniques proposed by different communities making use of the Web. The Semantic Web project is moving forward in becoming an important actor in the mainstream. Applications, tools, Semantic Web languages are constantly being developed creating a solid background for future Semantic Web developments and a valuable pool of experience are gained from the effort spent on these developments.

Hotel Search Using Semantic Web Technologies

The online travel segment was one of the first success stories in e-commerce, but since online travel agencies are facing fierce competition, to maintain a winning position or simply just to survive they do not only have to offer effective services and good value-for-money hotels or trips but most of all deliver more customized results w.r.t customer needs. The importance of quality of service depends on the different wishes of customers and the ever-changing demands of the market are paramount in today's world. The increasingly individualistic consumption patterns and lifestyles will make it increasingly difficult for tourist service providers to anticipate consumer behavior and configure their services accordingly, i.e. the tourist industry must focus more on a "hybrid consumer" whose travel choice will be increasingly complex. To tackle these problems head on and to give travel agencies, online portals and hotels the opportunity to take full advantage of the modern convenience of electronic business-to-business and business-to-customer trading. The general goal of such projects is to semantically connect, organize and share currently isolated pieces of tourism information in order to enable better teroperability and integration of inter- and intra-company travel information systems, facilitate the user to find and understand the information sources as well as to allow for individual use of travel offers. Better interoperability and integration of inter- and intra-company travel information systems, facilitate the user to find and understand the information sources as well as to allow for individual use of travel offers.

In the “Semantic Hotel Search” thesis we have concentrated on the usage of Semantic Web technologies to build a hotel recommendation engine which focuses on ranking the available hotels by matching customer profiles with hotel features, e.g. hotel or room amenities, by using additional information regarding points of interest as well as geographical dependencies. We use RDF as well as non-RDF data, but semantics play a vital role in our Semantic Hotel Search application in that it provides a shared and common understanding of data and services provided by travel information systems. Our engine assists in the various phases of the hotel search process [46]:

Information providers & hotel description: Nowadays hotel descriptions are written in free text – which limits the machine processability of matching hotel offers to customer’s needs. In contrast, the usage of a common language in the form of a set of controlled vocabularies (ontologies) to describe hotel details not only facilitates the communication and integration of individual hotels into the online travel portal but also improves the hit frequency (the more precise the hotel description the more often the hotel will come up).

Customers & their needs: In every travel portal customers (travelers and travel managers) can define their requirements either by filling out (“on the fly”) a request form or creating a user profile, which is then saved, or both. In each case the quality of the matches between the defined user needs and hotel characterization is directly proportional to the precision of both parties’ descriptions: semantic annotation using a common ontology, one that has a predefined set of vocabularies, enhances the accuracy of the match results.

Travel portal & the selection/ranking of the suitable offers: Besides improving the precision of the matching process another benefit from having hotel and user requirements annotated with terms from a controlled vocabulary is that the matching engine can utilize (additional to the hotel description) background knowledge such as point of interests or public transportation information: e.g. a physically disabled customer’s needs (information gleaned from user profile) will best be met by a hotel which has been specially outfitted for the disabled (information from hotel description) and is near a subway station which also has access to an elevator (information form public transportation data).

The purpose of this thesis is to explore the potential advantages of Semantic Web ontologies and to demonstrate how different technologies can be combined to create applications primarily based on ontologies.Our Semantic web application, developed within the thesis “Semantic Hotel Search”, provides semantic matching by combining data obtained from online reservation systems with ontology-based matching functions between hotel descriptions and customer profiles.

4.1. Overview of the Application

As explained in the previous chapters, there are several benefits of Web ontologies in the Semantic Web context. The developed Web application for this thesis is based on making use of Semantic Web technologies to show the benefits of such technologies. The overall structure of the system is illustrated in the Figure 4.1 below.

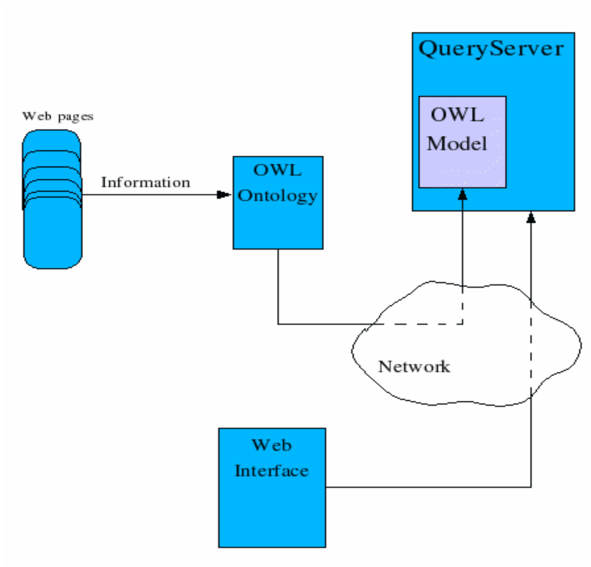


Figure 4.1 The Overall Structure of the Application

Current online hotel booking systems do not help travelers find hotels that meet their interests, so we have defined hotel-domain ontology to include three kinds of concepts:

1. Those associated with hotels themselves and their relationships with

neighboring artifacts

2. Those representing components of a hotel to be evaluated and their properties

3. Those corresponding to subjective, ambiguous terms that are used when searching or evaluating hotels. To demonstrate the use of the ontology for a hotel search, we developed a Semantic Hotel Search System using currently available semantic web technologies. Finally, we show how an interesting, but complex, hotel search query can be processed to find hotels using the system.

The application is mainly a web-based interface for accessing and querying content stored in an OWL ontology which can be located on the local system or on any Web server located somewhere on the Web. The targeted content domain is hotel information whose data/information has been collected from various Web sites giving access to hotel information.

The ontology being processed is not only intended to be used to retrieve data for various hotels but also to structure the general view and behavior of the Web interface such as categorizing the hotels and displaying them under a navigation menu. All the information and Web content presented at the front end of the application is extracted from the OWL ontology.

The actual ontology processing task is done by a different OWL server implemented for this thesis. The Web interface retrieves the necessary data from this server through a TCP connection. Loading and creating a model from OWL ontology, hotel category extraction, hotel querying and hotel content extraction is all being done by the OWL server working in the background of the application. As mentioned before, the Web interface is a separate module interacting with OWL server only to accept user input and present data returned from the server.

For constructing and creating the OWL ontology, a separate ontology editor has been used. The ontology constructing and managing part have been kept external for the thesis implementation because of the powerful editors already available today.

As functionality the Web interface provides an intuitive and easy to use interface allowing users to browse through the hotels and providing search capabilities so the users can easily find a hotel in order to book based on a certain specification.

4.2. Application Domain

As the information domain, hotel booking system has been selected for the thesis because of its various interesting properties. A number of characteristics of a hotel facilities and its environment such as golf, shopping center, swimming pool as well as the attributes of the trip context, such as distance to a meeting place or public transportation like airports, accessibility must be taken into consideration. Information about hotels are widely being presented on the Web on a large number of Web sites. This makes it easy to find information related to hotel and use these data when constructing the ontology.

Currently, tourism information systems provide good use cases for studying the potential brought by semantics and ontologies to solve the integration and interoperability problems they have been confronted with for many years. It is easy to classify the existing data and present this classification with an ontology. Concepts such as classes, subclasses, properties and relations can easily be applied and demonstrated within this domain.

The constructed ontology contains a large number of hotel information in which data has been collected from the currently existing Web sites provide to book hotels. Because of the common information structure of the hotels it has been easy to create a common format and structure to store these information about hotels in the ontology file that has been constructed. Properties such as rooms, stars, locations, facilities, activities, nearest tourist attractions information etc. are all common to hotels. In addition, the different features are all common to the domain, allowing them to be reusable definitions instead of being distinct to each and every hotel.

booking.com was used to obtain hotel information. booking.com is a Dutch online booking portal. It was established in 1996, and offers accommodation booking. It claims to deal with more than 550,000 room nights' reservations per day. The main reason to use this web site is easy to parse information about hotels and simple to categorize features in order to apply on the ontology. Also, it has extensive hotel network and hundreds of reviews about hotels the combination of both a very positive score and a large number of reviews certainly delivers very powerful (and persuasive) social proof of some of the hotels.

The second hotel website that we used for obtain information is *tripadvisor.com.* It is a travel website providing directory information and reviews of travel-related content. It also includes interactive travel forums. Its substructure is not as convenient as booking.com in order to obtain facilities of hotels so it was used for hotel review's sentimental analysis.

Including tripadvisor and booking, it has not been possible to find Web sites which have published their hotel contents in any form of an ontology. All the available data was marked up with HTML (Hyper Text Markup Language) which makes it almost impossible for other systems to access the data and make use of it effectively.

When providing data for the ontology constructed for the application, automated process have been created and used. Extracting information from such sites has been done by creating html parser and retrieving the hotel information from Booking.com and Tripadvisor.com. This part of thesis is explained in detail in Chapter 5.

Before designing the system, a detailed investigation of the different web sites has been performed. The main focus was on the capabilities related on how different users can access the relevant content as fast as possible. Not one of the mentioned web sites have advanced search capabilities except for some of them which allows hotel search based on the facilities. Some of the sites provide search capabilities only based on keyword search where only the hotel titles are being used for keyword matching.

The content stored at various Web sites is not structured in a way so that they are accessible for other systems. That is, the content is not reusable and is just waiting for users which have plenty of time and passion to seek for it. The only way of retrieving the stored information is by manually copying and pasting the information so that it can be used for other purposes.

Storage and representation of information: The domain information is represented with an ontology. All the data related to hotel domain including classifications, properties and relations are all being stored in the ontology file. Tdb is used to store RDF files and all information about hotels.

Ontology language: The ontology language used to construct the ontology was specified as OWL because it is currently the most powerful ontology language with a greater representational power compared to other ontology constructing language. The OWL sublanguage has been specified as OWL DL because some of the more advanced class related constructs such as subClassOf and disjointWith was used in constructing the ontology.

Ontology processing: Ontology processing is being performed by using the semantic web framework Jena. Ontology processing is implemented by making use of the API provided by Jena and some crud operations are realized using jena library.

Web interface: The Web interface is a Web-based application that handles the visual presentation of the hotel contents and provides navigation through the different categories of hotels. It provides an easy to use search interface allowing the users to construct queries with different criteria.

Application development platform: The OWL server responsible of ontology processing has been developed using the popular object oriented programming language ASP.NET. The Jena API library was optimized for .Net The Web interface is implemented using the widely used C#, JavaScript and jquery.

4.3. Application Design

The Ontology-driven hotel search engine application developed for this thesis is built up in five main parts; the OWL ontology, Web interface, semantic services, sentimental analysis and Html parser to interact with the system. The ontology constructed is the only information resource used for the application. All data such as the text representing the hotel, facilities, review etc. are stored in the ontology file constructed. Even the link names appearing on the menu displayed on the Web interface are stored and retrieved from the ontology file.

After built ontology, needed to feed with hotel data. In order to achieve this html parser is created. Taking a given URL as parameter, it parse hotel facilities, specifications, reviews from Booking.com and Tripadvisor.com then load to tdb. Two services were developed. One of them is windows service in order to update hotel reviews every day. The other one is web services that acts as a bridge between the constructed ontology and the web interface. It provides crud operations. After constructed OWL ontology and data, it is ready to accept requests from the Web interface.

Mainly four types of requests can be made from the Web interface.

- Request for hotel categories, countries, cities.
- Request for all city attractions under a specific country.
- Request for a hotel given a specific resource id for the hotel.
- Perform a search given some query.

The web interface is a .Net application which only performs the user interaction with the OWL ontology server/model. The Web interface does not deal with any data processing other that making requests to the OWL server and presenting the responses in an HTML formatted page. It is responsible of accepting user input, creating a request message and sending it to the OWL server. When the server returns the corresponding response, the Web interface simply displays it to the user. The interface is available selective facilities, specialties and the hotel information being displayed are all retrieved from the OWL server dynamically on each page request. Whenever the category structure and the content of the ontology file has been changed and modified, all the changes are reflected at the Web interface and are made available to the user without making any modification to the code for the Asp.Net web application.

After users select the hotel, they can see the hotel details based on their criteria. For example user selects criteria like swimming pool, bar and fitness then hotels which have these facilities are listed. After the selection one of them, in detailed page user finds more details about these criteria like as positive and negative comments from people who stayed there. Sentimental analysis part take charge in here. In addition to, users can find most positive features and negative features about hotel.

The communications between the different parts of the system are based on commonly used network communication techniques. All the five parts of the system can be located remotely or on any machine having access to the Internet.

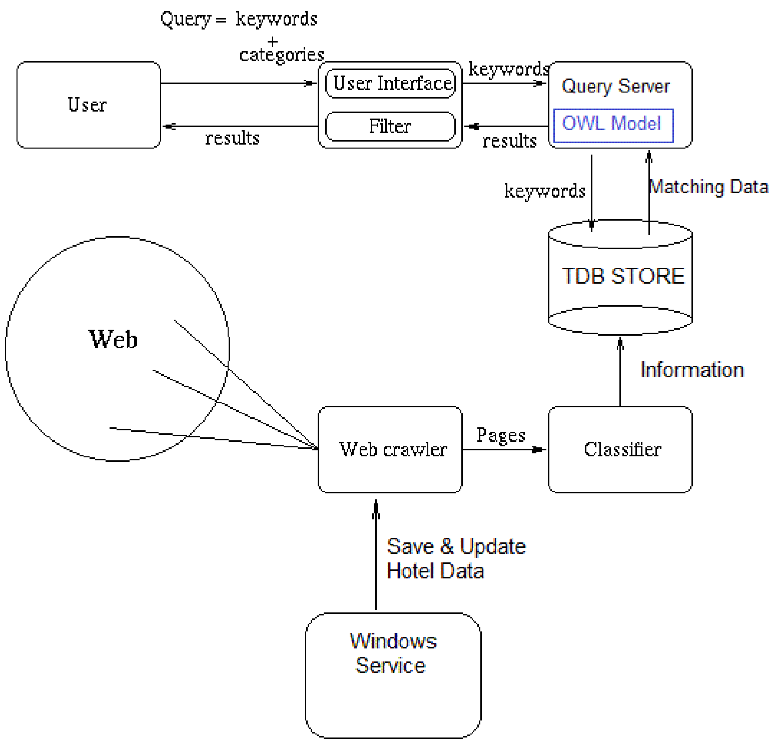


Figure 4.2 Architecture of Semantic Hotel Search Application

Protégé Ontology Editor: Protégé is developed by Stanford Medical Informatics. It has been selected as the ontology editor for the thesis because it is suitable to the project in many ways [47]. It is a widely used free ontology editor especially for constructing and maintaining OWL ontologies. OWL is different than other ontology languages in that it supports a richer set of operators such as AND, OR, and Negation. Protégé supports all the advanced properties of OWL language and provides all the functionality to maintain OWL ontologies. Because of its wide popularity, it was easy to obtain support for it.

Plenty of tutorials, documentation and support forums are available for the editor. The functionality of Protégé can be extended by various plug-in available online on the home page for Protégé. In addition, different wizards to ease the work are provided. The internal logical model allows it to interact reasoning services for example to compute inferred types and make consistency checking on the ontology being constructed.

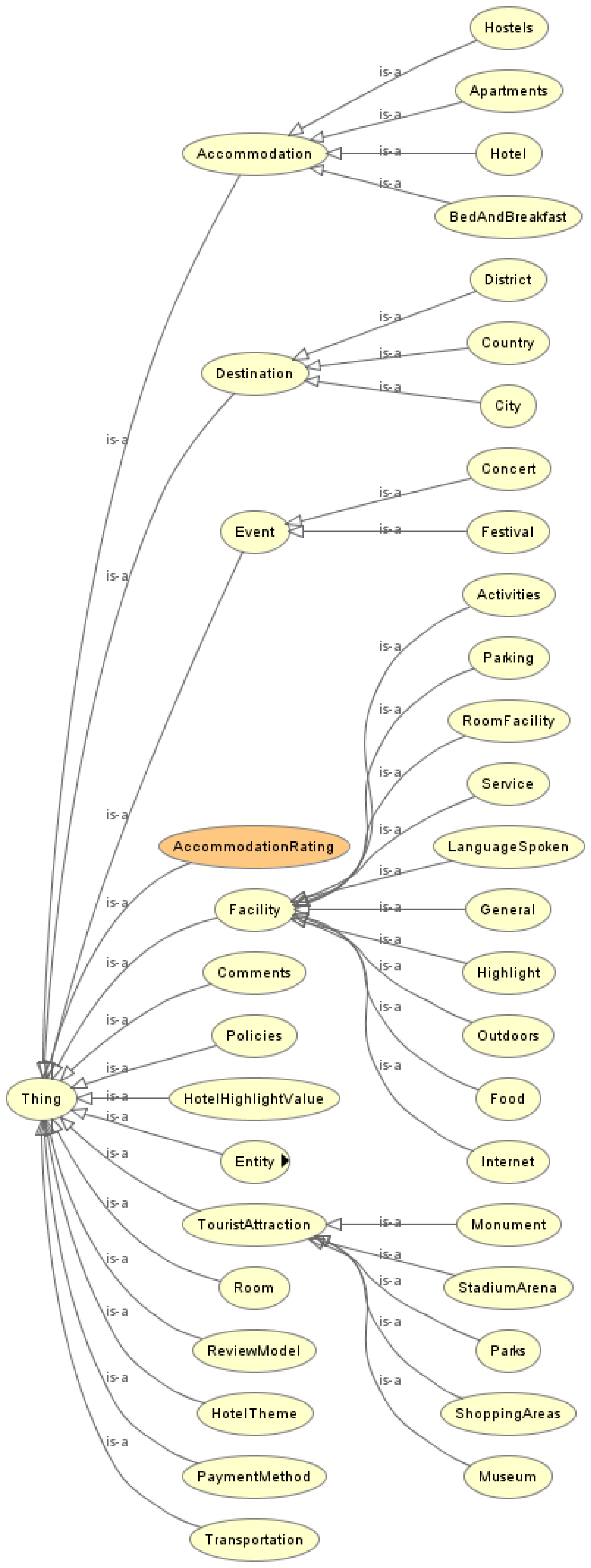
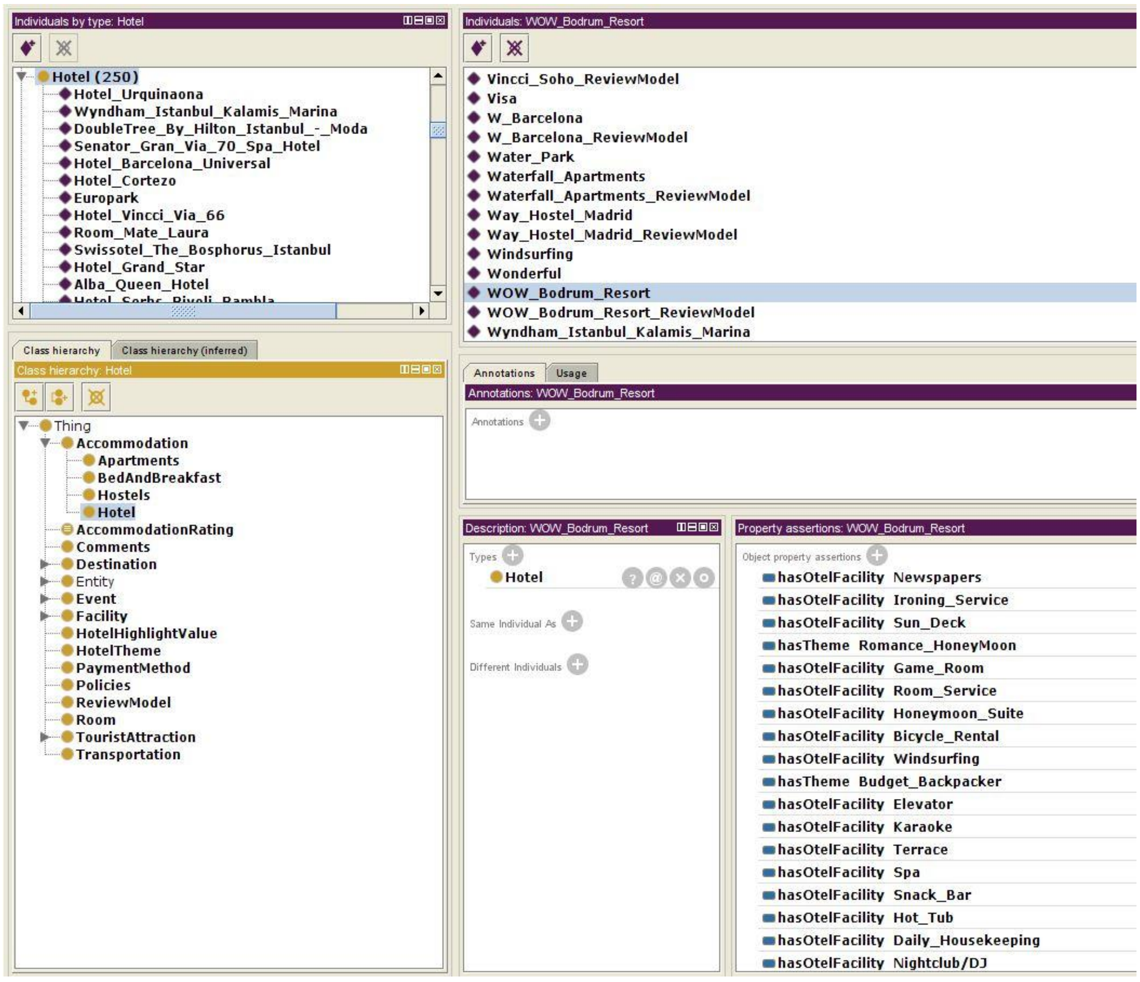


Figure 4.3 Classes of Hotel Domain Ontology


Individuals by type: Hotel

- Hotel (250)
  - Hotel_Urquinaona
  - Wyndham_Istanbul_Kalamis_Marina
  - DoubleTree_By_Hilton_Istanbul_-_Moda
  - Senator_Gran_Via_70_Spa_Hotel
  - Hotel_Barcelona_Universal
  - Hotel_Cortezo
  - Europark
  - Hotel_Vincci_Via_66
  - Room_Mate_Laura
  - Swissotel_The_Bosphorus_Istanbul
  - Hotel_Grand_Star
  - Alba_Queen_Hotel

Class hierarchy | Class hierarchy (inferred)

Class hierarchy: Hotel

- Thing
  - Accommodation
    - Apartments
    - BedAndBreakfast
    - Hostels
    - Hotel
  - AccommodationRating
  - Comments
  - Destination
  - Entity
  - Event
  - Facility
  - HotelHighlightValue
  - HotelTheme
  - PaymentMethod
  - Policies
  - ReviewModel
  - Room
  - TouristAttraction
  - Transportation

Individuals: WOW_Bodrum_Resort

- Vincci_Soho_ReviewModel
- Visa
- W_Barcelona
- W_Barcelona_ReviewModel
- Water_Park
- Waterfall_Apartments
- Waterfall_Apartments_ReviewModel
- Way_Hostel_Madrid
- Way_Hostel_Madrid_ReviewModel
- Windsurfing
- Wonderful
- WOW_Bodrum_Resort
- WOW_Bodrum_Resort_ReviewModel
- Wyndham_Istanbul_Kalamis_Marina

Annotations | Usage

Annotations: WOW_Bodrum_Resort

Annotations

Description: WOW_Bodrum_Resort

Types
Hotel

Same Individual As

Different Individuals

Property assertions: WOW_Bodrum_Resort

Object property assertions

hasOtelFacility	Newspapers
hasOtelFacility	Ironing_Service
hasOtelFacility	Sun_Deck
hasTheme	Romance_HoneyMoon
hasOtelFacility	Game_Room
hasOtelFacility	Room_Service
hasOtelFacility	Honeymoon_Suite
hasOtelFacility	Bicycle_Rental
hasOtelFacility	Windsurfing
hasTheme	Budget_Backpacker
hasOtelFacility	Elevator
hasOtelFacility	Karaoke
hasOtelFacility	Terrace
hasOtelFacility	Spa
hasOtelFacility	Snack_Bar
hasOtelFacility	Hot_Tub
hasOtelFacility	Daily_Housekeeping
hasOtelFacility	Nightclub/DJ

Figure 4.4 Sample individuals with property values on Hotel Ontology

Sentigem API: Sentigem is a platform that offers an easy-to-use sentiment analysis tool for English language based documents or text blocks. To this end, we provide an API endpoint for computing sentiment from various consumption points determined by the user and passed to our API as text [48] Sentigem api is used for analyzing reviews of hotels. The service is presently an English only service and The Sentigem API service only supports HTTPS calls. It is free api but needs the key. After sign up, the key is given. Sentigem takes key and text as a parameter then returns polarity of the text as positive, negative or neutral.

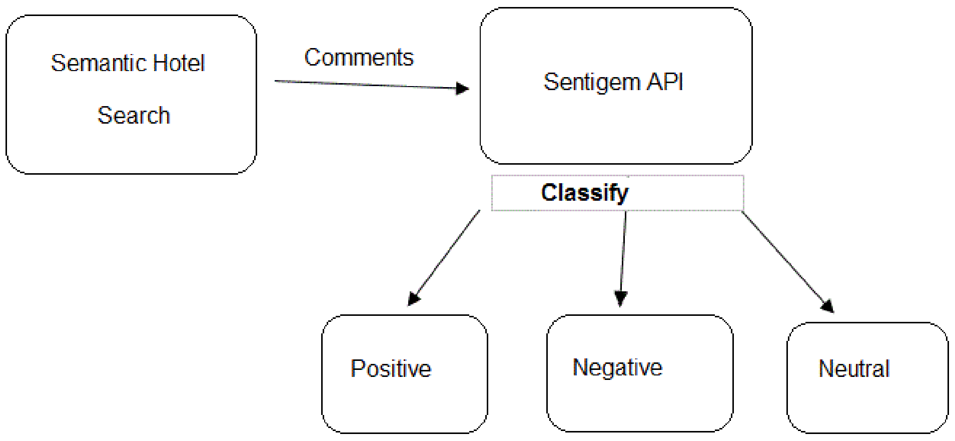


Figure 4.5 Sentiment Analysis Architecture of Semantic Hotel Search Application

CHAPTER 5

IMPLEMENTATION

In this section we describe our prototypical implementation of the technical infrastructure of our Semantic Hotel Search by presenting a solution for data integration, semantic matching and evaluators. Our hotel evaluation engine is composed of five main parts: the OWL ontology, web interface, semantic services, sentimental analysis and Html parser to interact with the system. All of them are implemented with different technologies

5.1. Implementing the Ontology with Protégé

The web ontology file has been created so that it will represent hotel domain ontology in detail. However, the ontology has not been created too fine grained, that it will be difficult to provide specific information for all of the details being built into the ontology. For example, room price according to searched date is not provided.

The ontology constructed makes advantage of concepts such as class hierarchies, class relations and properties. In order to model the information domain in a realistic manner, proper information description methods have been used together with the powerful logic descriptors provided by OWL-DL and OWL-Full. The following part will explain the different classes that have been defined together with the different relations and properties. Domain ontologies specify concepts, relationship among concepts, and inference rules for a single application domain (e.g., airline reservations, art galleries, furniture, fishing, gourmet food) or task. Our domain is hotels.

Ontology creation requires heuristics and expertise, rather than an engineering approach. Prior research has concentrated on related tasks, such as ontology learning, ontology evaluation, evolution, and merging For example, ontology learning seeks to discover ontological knowledge from various forms of data automatically or semi-automatically using methods and tools such as UIMA, GATE, OpenCalais, and WikiOnto. Several ontology creation methodologies have been proposed. There are six-step methodology for semi-automated ontology creation using terms from the World Wide Web. “The methodology is heuristic in nature and takes advantage of existing tools. The methodology is based on a framework for ontology learning proposed by Zhou (2007) and METHONTOLOGY (Fernández-López et al., 1997), which provides high-level steps for ontology creation as an existing partial solution. Our research expands and augments prior work and integrates tools. [49]" Figure 18 provides an overview of our methodology.

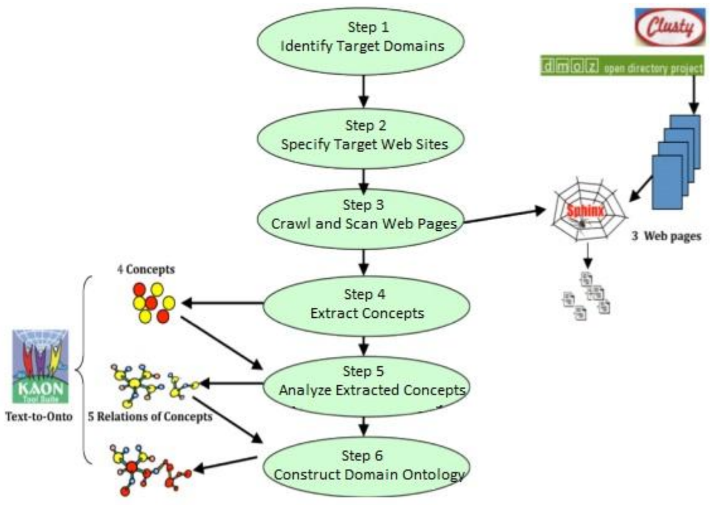


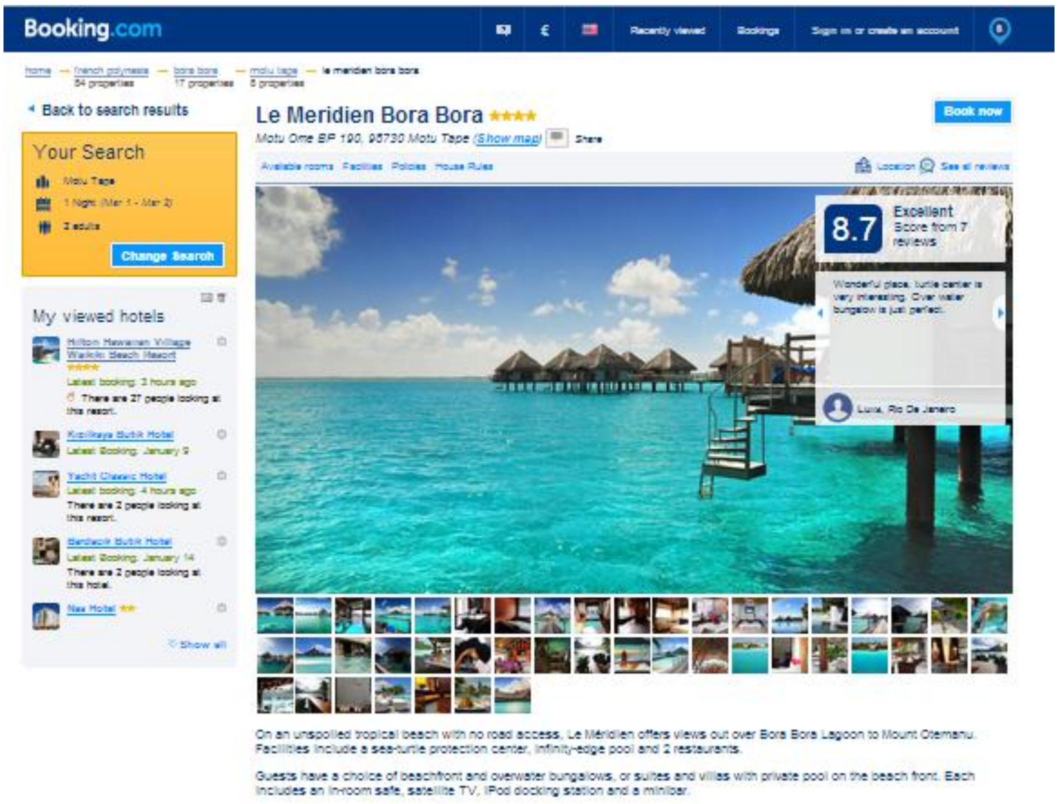
Figure 5.1 Methodology for Domain Ontology Creation[14]

Application domains for which ontologies are needed may be of various sizes. Therefore, the initial step in domain ontology creation is to determine the scope of the application domain (e.g., food recipe, sports, wine, pizza, automobile, hotel etc...)

This step is driven by the reason for the ontology development, intended uses, and potential users. The extent requires the identification of the categories of the domains in which the users are interested. Several initiative have been made to categorize the many and diverse web pages on the www into domains motivated by the fact that search engines are often unable to provide content-dependent, useful results.

The second step is determine target web sites. Target web sites are specified within a domain or category selected from Step 1. The purpose of this step is to provide the basic resources for the next four steps. booking.com and tripadvisor.com are the biggest and the most preferred web sites for searching and booking accommodations.

[14] Kim W. j., "*A Novel Approach to Ontology Management*" (ScholarWorks., Georgia State University, 2010), 146


Booking.com
Recently viewed
Bookings
Sign in or create an account
home — french polynesia 84 properties — bora bora 17 properties — motu tape 5 properties — le meridien bora bora
Back to search results
Le Meridien Bora Bora ★★★★
Motu Ome BP 190, 98730 Motu Tape (Show map) Share
Book now
Available rooms Facilities Policies House Rules
Location See all reviews
Your Search
Motu Tape
1 Night (Mar 1 - Mar 2)
2 adults
Change Search
8.7 Excellent Score from 7 reviews
Luiza, Rio De Janeiro
My viewed hotels
Hilton Hawaiian Village Waikiki Beach Resort
Kizilkaya Butik Hotel
Yacht Classic Hotel
Bardacik Butik Hotel
Nas Hotel
Show all

Wonderful place, turtle center is very interesting. Over water bungalow is just perfect.

Latest booking: 3 hours ago
There are 27 people looking at this resort.

Latest Booking: January 9

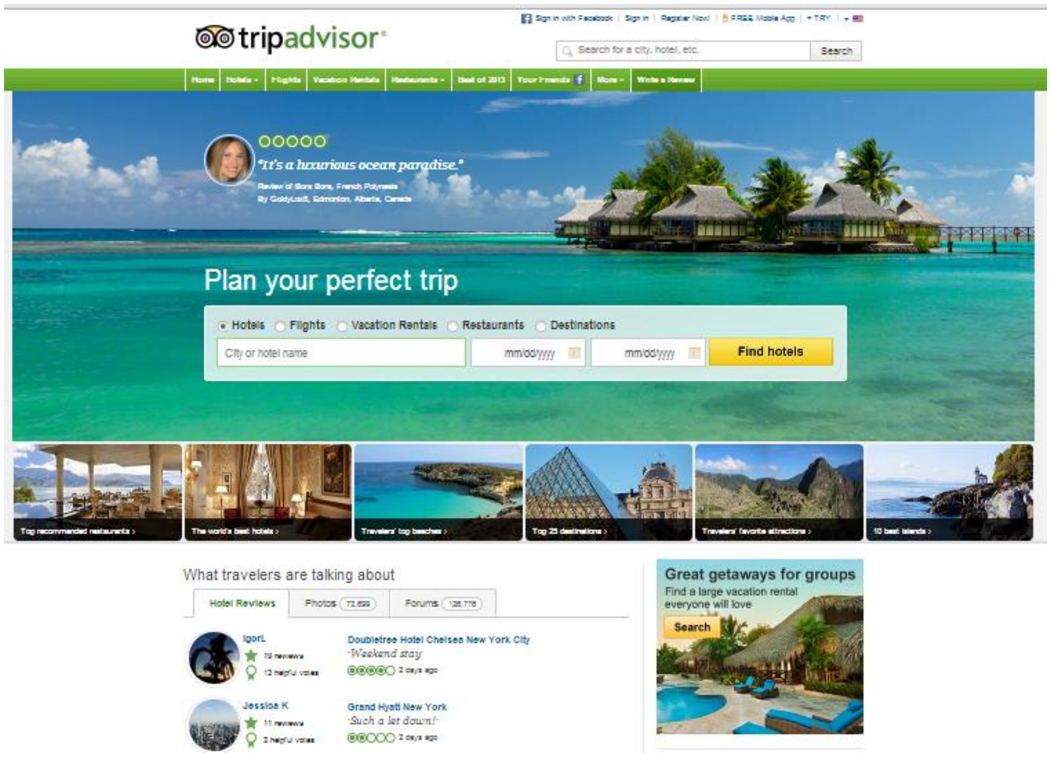
Latest booking: 4 hours ago
There are 2 people looking at this resort.

Latest Booking: January 14
There are 2 people looking at this hotel.

On an unspoiled tropical beach with no road access, Le Méridien offers views out over Bora Bora Lagoon to Mount Otemanu. Facilities include a sea-turtle protection center, infinity-edge pool and 2 restaurants.

Guests have a choice of beachfront and overwater bungalows, or suites and villas with private pool on the beach front. Each includes an in-room safe, satellite TV, iPod docking station and a minibar.

Figure 5.2 booking.com



"It's a luxurious ocean paradise."
Review of Bora Bora, French Polynesia
By GoldyLoc8, Edmonton, Alberta, Canada

What travelers are talking about

IgorL
19 reviews
13 helpful votes
Doubletree Hotel Chelsea New York City
"Weekend stay"
2 days ago

Jessica K
11 reviews
3 helpful votes
Grand Hyatt New York
"Such a let down!"
2 days ago

Figure 5.3 tripadvisor.com

Step 3 is crawling and scan web pages. Now we can crawl hotel information from the websites. The way of crawling data is explained in detail in next section 5.2. Html Parser. The needed information about hotels are main features and specialties hotels and some concepts associated with hotels themselves and their relationships with neighboring artifacts such as airports, subway stations, attractions and museums.

Candidate terms for ontologies are selected at step 4 by user. The user browses the results of step 3 and selects candidate terms related to the domain based upon the importance and the relevance of the term to the domain ontology.

The purpose of step 5 is to analyze terms and identify relationships among selected terms.

Last and final straight forward step is constructing the domain ontology. Ontology was created an implemented using the help of the powerful tools and wizards the Protégé development environment provides. It allows creating and populating classes and concepts by simple clicks while providing a visual overview of the created classes and instances with class hierarchies and instance diagrams. Creating properties and relations (objectProperties, dataProperties) is done similarly by defining the name of the property and specifying the domain-range relation of the constituent elements/definitions.

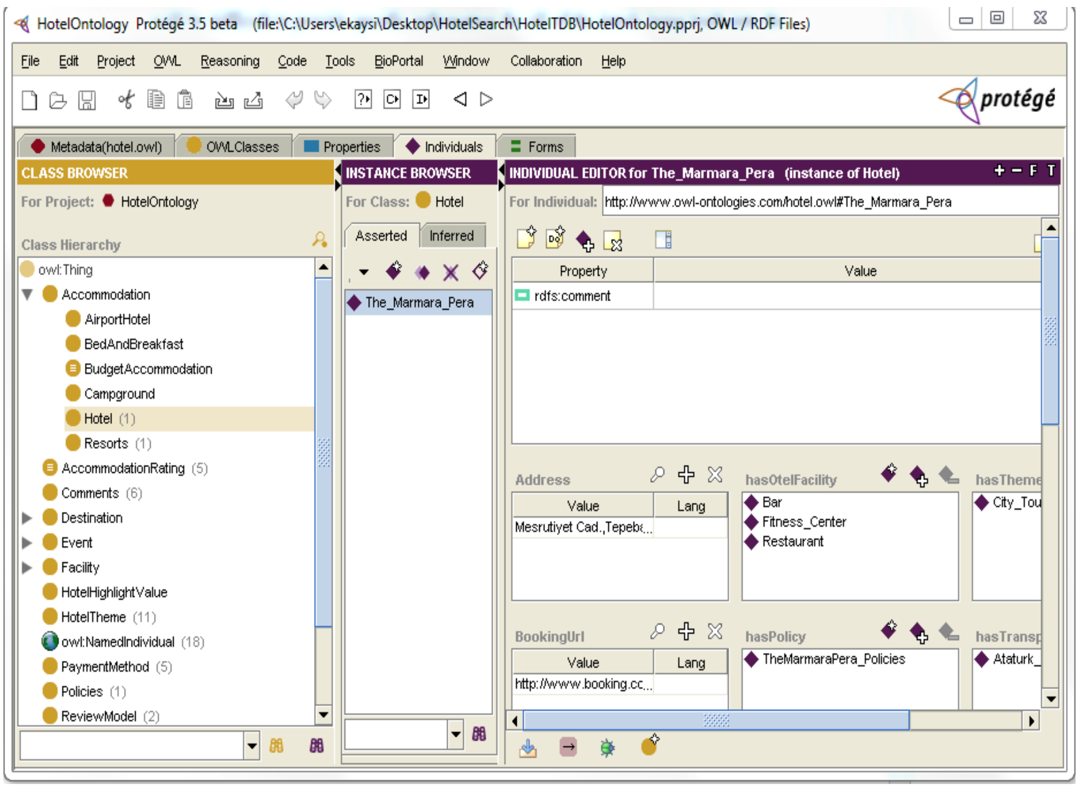


Figure 5.4 Protégé Ontology Editor

OWL Classes:

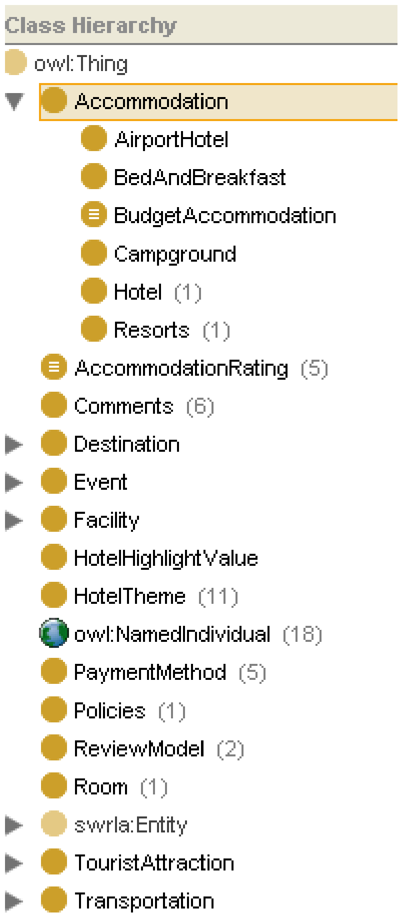

Class Hierarchy
owl:Thing
- Accommodation
  - AirportHotel
  - BedAndBreakfast
  - BudgetAccommodation
  - Campground
  - Hotel (1)
  - Resorts (1)
- AccommodationRating (5)
- Comments (6)
- Destination
- Event
- Facility
- HotelHighlightValue
- HotelTheme (11)
- owl:NamedIndividual (18)
- PaymentMethod (5)
- Policies (1)
- ReviewModel (2)
- Room (1)
- swrla:Entity
- TouristAttraction
- Transportation

Figure 5.5 Hotel Domain Ontology Class hierarchy

Accommodation: This class is about accommodation types. It contains 6 subclass and at the same time these are used as the search criteria in the web interface.

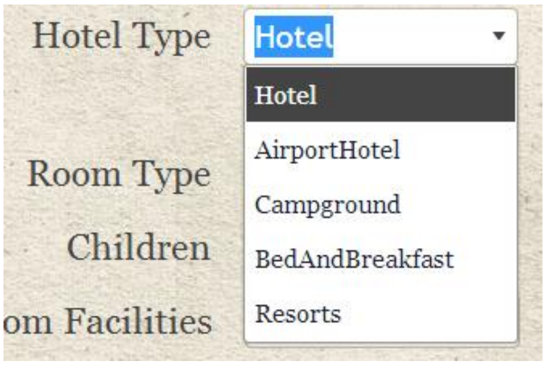


Figure 5.6 Hotel Type Categories

Accommodation class has object and data type property. Datatype properties are Name, Description, Price, Address, BookingUrl and TripadvisorUrl.

Object properties are;

hasDestination: relationship with Destination class. Defines which district the place of the hotel is.

hasStar: relationship with AccomodationRating class. Defines how many star hotel has.

hasComment: relationship with Comments. It determine hotel review score from Booking and tripadvisor.

hasTransportation: relationship with Transportation class. It specify how to access to hotel such as plane, bus, ship etc…

hasOtelFacility: relationship with OtelFacility class. It describes what facilities hotel have.

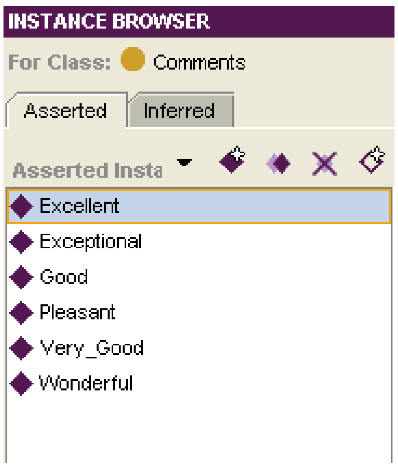
hasPolicy: relationship with Policy class. It sets hotel's policy rules.

hasReviewModel: relationship with ReviewModel class. It set hotel's review based on polarity like good or bad review.

hasTheme: relationship with HotelTheme class. It defines hotel theme such as luxury, spa, mountain, seaside etc…

Accommodation Rating: It specify hotel stars. It has five individuals (one, two, three, four and five stars). It has data property which names "star".

Comments: Comments class remark hotel's score over ten that is taken by Tripadvisor and Booking users. It has data property which names "score".



Excellent
Exceptional
Good
Pleasant
Very_Good
Wonderful

Figure 5.7 Comment Individuals.

The score schema for comment class' individuals is below:

Caption	Score
Exceptional	9.5-10
Wonderful	9.0 - 9.5
Excellent	8.5 - 9.0
Very Good	8.0 - 8.5
Good	7.0 - 8.0
Pleasant	6.0 -7.0

Table 5.1 Score Schema of Comments Class Individuals

Destination: It specifies the place of hotel. It has three subclasses: district, city, country.

There is object property named "isDestinationOf" between Accommodation and Destination. "IsdestinationOf" and "hasDestination" properties are inverse. In addition to, City class has these object relations:

belongsToCountry: implies the city which is in a specified country.

hasAttraction: specify tourist attraction places in the city.

hasEvent: specify events such as concerts, festivals etc... in the city.

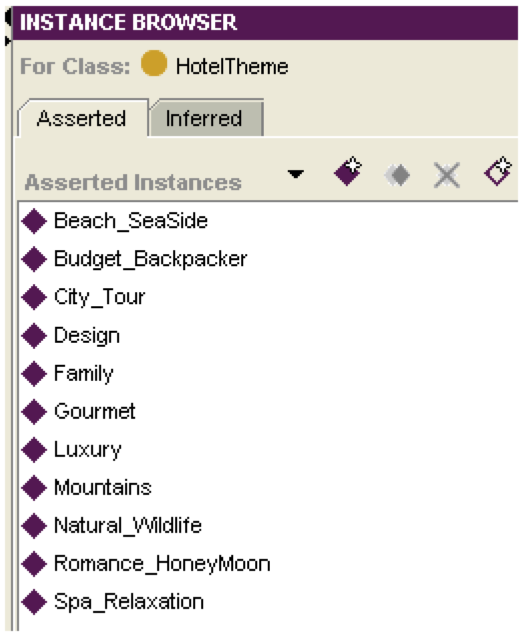
Country class has data property that is "shortName" and District class has object property that is "belongsTocity" represent which city of the districts belongs to.

Event: This class represent events such as concerts, festivals, special carnival etc… Event class has two subclass concert and festival.

Facility: Facility class about hotel features and specialties. It has these subclasses: activities, food, highlight, Internet, room facility, language spoken, outdoors, parking, general and service. Dataproperty of this class is "name".

HotelHighlightedValue: This class implies polarity and rates of the most popular features of the hotels. Its data property is "polarity" and "rate". Also its object relation is with "Highlight" class.

HotelTheme: It indicate theme of the hotels. It contains individuals shown on the below figure:


INSTANCE BROWSER

For Class: HotelTheme

- Beach_SeaSide
- Budget_Backpacker
- City_Tour
- Design
- Family
- Gourmet
- Luxury
- Mountains
- Natural_Wildlife
- Romance_HoneyMoon
- Spa_Relaxation

Figure 5.8 Instance of Hotel Theme Class

PaymentMethod: It refers to payment methods to hotel charge. Payment method class instances like: AmericanExpress, visa, master card, PayPal, cash.

Policies: It defines some general hotel rules. It has data property like allowed pets, checking, checkout. Also it has relationship that named “hasPayment” with “PaymentMethod”.

ReviewModel: It contains reviews, good and bad facilities of hotel. Datatype property of this class is “review” and also it has object property that named “hasBadFacility” and “hasGoodfacility” with “HotelHighlightValue” class.

Room: It defines hotel’s room and room facilities. It has data property like “name” and “room size” and has object property “hasRoomFacility” with the “RoomFacility” class.

TouristAttraction: This class specify touristic places like Eiffel tower, The Topkapı Palace etc... It has subclasses: monument, parks, museums, shopping areas, and stadium arena.

Transportation: Transportation class implies how to access accommodation place and which transportation vehicle is close to hotel. It has subclasses: airport, bus terminal, port, train station.

Please refer to the Appendix – A for hotel domain ontology owl file and Appendix B for hotel domain Onto Graf visualization.

5.2. Html Parser

The current web provides data that is only roughly connected and cannot be used for rich querying. Especially tourism area is very dynamic because it is one of the world's largest industries and its growth shows an increase every year. Also, tourism are is not just be formed with Hotels there are other related areas such as renting car, transportation to hotel etc.. In our thesis we need these information but we cannot use directly these of information because of lack of availability of current web technologies. In addition, it has not been constructed enhanced ontology in order to use ontology with their instances. Because of these reasons, we need to extract information from web pages using some web parsing tools. In order to parse web pages , HtmlAgilityPack tool is used.

HtmlAgilityPack is an agile HTML parser that builds a read/write DOM and supports plain XPATH or XSLT It is a .NET code library that allows you to parse "out of the web" HTML files [50].

Semantic hotel domain data are supplied from www.booking.com. The main reason to use Booking that this website has the closest information to ontology structure. Before parsing Booking, we first to do is that choice hotels which we want to add to our Tdb. This is the most troubled, challenging and time consuming process. I created "Hotel.xml" file to store chosen hotel data in order to make other process is automatic and dynamic. Extracting hotel information from url and saving to tdb are quite dynamically created. The xml schema developed in this thesis is on the next page.

```
     <?xml version="1.0" encoding="utf-8"?>
     <HOTELS>
       <Hotel>
       <bookingUrl>http://www.booking.com/hotel/ad/roc-de-
caldes.en-
us.html?sid=ef88f56814d252602150c679d9dca258;dcid=1;checkin=2013
-12-28;checkout=2013-12-
29;srfid=68fc5fb87b85b21c7550b91b6d26ccce11224226X1</bookingUrl>

<tripadvisorUrl>http://www.tripadvisor.com/Hotel_Review-g190401-
d508095-Reviews-Hotel_Roc_de_Caldes-
Les_Escaldes_Escaldes_Engordany_Parish.html</tripadvisorUrl>
        <type>Hotel</type>
        <theme>City_Tour</theme>
        <img>http://r-
ec.bstatic.com/images/hotel/max300/199/19900735.jpg</img>
        <room>Standard Guest Room</room>
        <room>Superior Guest Room</room>
```

```
      <room>Deluxe Class Room</room>
      <price>245</price>
      <transportation>Bus_Terminal</transportation>
      <district>Escaldes-Engordany</district>
      <city>Andorra la Vella</city>
    </Hotel>
  </HOTELS>
```

Hotel.xml file stores data which we cannot extract from web pages like hotel type, price, and room types etc… the element of xml "bookingUrl" implies URL of chosen hotels. Also bookingUrl and tripadvisorUrl are used for extracting hotel reviews.

After creating xml schema file, we ready to extract information from the hotel web page. Firstly all xml nodes are read and for each node, we take bookingurl element value then sends it to GetHtmlPage method as parameter. Its code example is in the below:

```
      XmlDocument doc = new XmlDocument();
doc.Load(HttpContext.Current.Server.MapPath(string.Format("~\\Ap
p_Data\\hotel.xml")))
      XmlNodeList nodes =
doc.DocumentElement.SelectNodes("/HOTELS/Hotel");
      foreach (XmlNode node in nodes)
      {
            string bookingUrl =
node.SelectSingleNode("bookingUrl").InnerText;
            string tripadvisorUrl =
node.SelectSingleNode("tripadvisorUrl").InnerText;
      }
```

After get hotel url from xml, we need to get full web page from url for this step our "GetHtmlPage" method takes hotel url as a parameter and make http request and returns full content of webpages as a string. Code sample is shown in the below code snippet.

```
       String strResult;
       WebResponse objResponse;
       WebRequest objRequest = HttpWebRequest.Create(strURL);
        objResponse = objRequest.GetResponse();
       using (StreamReader sr = new
StreamReader(objResponse.GetResponseStream(),
Encoding.GetEncoding("UTF-8")))
       {
          strResult = sr.ReadToEnd();
          sr.Close();
       }
        return strResult;
```

Then this string content converted to HtmlDocument type by HtmlAgilityPack .

```
        string response = GetHtmlPage(bookingUrl);
        HtmlDocument dokuman = new HtmlDocument();
        dokuman.LoadHtml(response);
```

Now, we acquired HtmlDocument file to parse information according to html tag id etc… For example we acquire hotel name, we have to investigate html tag id, class names etc… "hp_hotel_name" is id of hotel title label.

```
        stringName=dokuman.DocumentNode.SelectNodes("//span[@id='h
p_hotel_name']").First().InnerText.Trim();
```

All these steps are repeated for each hotel node. Each hotel created as individual for belong to class and set relationships between classes using semantic web services. This semantic web services part is explained in section 5.3. Semantic Services. In addition to, it is not enough to create ones all information then use it. We need new updated information hotels and reviews. For these reasons, windows service was created to update all information on every day. The detail code snipped about html parser is in the below:

```
        XmlDocument doc = new XmlDocument();
        doc.Load(HttpContext.Current.Server.MapPath(string.Format(
"~\\App_Data\\hotel.xml")));
        XmlNodeList nodes =
doc.DocumentElement.SelectNodes("/HOTELS/Hotel");
        foreach (XmlNode node in nodes)
        {
        string bookingUrl =
node.SelectSingleNode("bookingUrl").InnerText;
                    string tripadvisorUrl =
node.SelectSingleNode("tripadvisorUrl").InnerText;
                    string response = GetHtmlPage(bookingUrl);
                    HtmlDocument dokuman = new HtmlDocument();
                    dokuman.LoadHtml(response);

            string Name =
        dokuman.DocumentNode.SelectNodes("//span[@id='hp_hotel_nam
e']").First().Inn
        erText.Replace("&amp;", "").Trim();
            string hasStar =
dokuman.DocumentNode.SelectNodes("//*[contains(@class,'use_sprit
es
        stars')]").Descendants().ParentNode.Attributes["title"].Va
lue;

        }
```

5.3. Semantic Web Services for Semantic Hotel Search Application

Web services add a new level of functionality to the current Web, transforming the Web from a distributed source of information to a distributed source of functionality. They provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. [51]

Semantic Web services as Web services in which Semantic Web ontologies ascribe meanings to published service descriptions, so that software systems representing prospective service clients can interpret and invoke them. Enriching Web services with semantic information allows automatic location, composition, innovation, and interoperation of services.

5.3.1. HotelService

As mentioned above, semantic hotel domain ontology was created. Hotel.xml file was created to build instances. Then, Hotel Individuals are retrieved from each hotel web pages. However it is not adequate to save individuals once, in order to ensure consistency and accuracy of information. Also up to date data is necessary for accurate reasoning about reviews and hotel. Therefore it is in need of a windows service to update hotel domain individuals.

Hotel Service thesis is a small windows service project inside Semantic Hotel Search web application. This service runs every day at 00:00 automatically and update all hotel facilities, specialties and reviews. The figure below shows some code snipped about service.

```
protected override void OnStart(sitring[] args)
{
        try
        {
                TimeSpan alertTime = new TimeSpan(00, 00, 00);
                DateTime current =DateTime.Now;
                TimeSpan timeToGo = aletTime - current.TimeOfDay;

                if (timeToGo.milliseconds < 0)
                {
                        DateTime dt = DateTime.Today.AddDays(1);
                        DateTime dt2 = DateTime.Now;
                        timeToGo = dt-dt2;
                        AddUpdateHotelReviewModel();
                }
                System.Threading.Timer timerforSourceRecord = new System.Threading.Timer (x =
                {
                        this.AddUpdateHotelReviewModel();
                }, null, timeToGo, new TimeSpan(24, 00, 00));
        }
        catch (Exception e)
        }
                WritetoLog(e.Message);
        }
}
```

5.3.2. Semantic Hotel Web Service

Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Applications access Web services via ubiquitous Web protocols and data formats such as HTTP, XML, and SOAP, with no need to worry about how each Web service is implemented. Web services combine the best aspects of component-based development and the Web, and are a cornerstone of the Microsoft .NET programming model. [52]. Because of this feature of webservice, developed semantic hotel webservice application.

Semantic Hotel Webservice was created in .net platform to interact web interface and windows service. This two separate application uses methods of web service. Although we can use these methods via creating it in both two application, it's useless and effortful because if something changed, it is needed to alter both application. It is time consuming.

Semantic Hotel Webservice provides methods for create, update, delete, get individuals from tdb. Also there is a method “RunQuery”. It takes a sparql text as parameter and runs this query then returns results as a datatable.

When this service starts, initially load ontology model from directory schema. It checks it was created before in another means if exits “GOSP.DATA”, populate ontologies else creates directory schema. Relevant sample codes are on the below.

```
      private string owlPath =
@"C:\Users\ekaysi\Desktop\HotelSearch\HotelTDB\";
      private string directorySchema =
"C:\Users\ekaysi\Desktop\HotelSearch\HotelTDB\tdbSchema";

      string path = Path.Combine(directorySchema,
"GOSP.DAT");
        if (!System.IO.File.Exists(path))
         {
               CreateOntologyDbSchema();
         }
         private void CreateOntologyDbSchema()
          {
                tdb = TDBFactory.createNamedModel("rbt",
Path.Combine(owlPath, "tdbSchema"));
                  FileManager.get().readModel(tdb,
Path.Combine(owlPath, "HotelOntology.owl"),
      "RDF/XML");
         }
```

Main semantic hotel webservice’s methods and definitions are on the below:

RunQuery: It takes a string sparql query and then returns results data table.

CreateIndividual: takes individual name, properties (object or data property) and class name as a parameter. Saves individual to dependent class.

UpdateIndividual: Individual name and properties are parameters. Alter properties of individuals

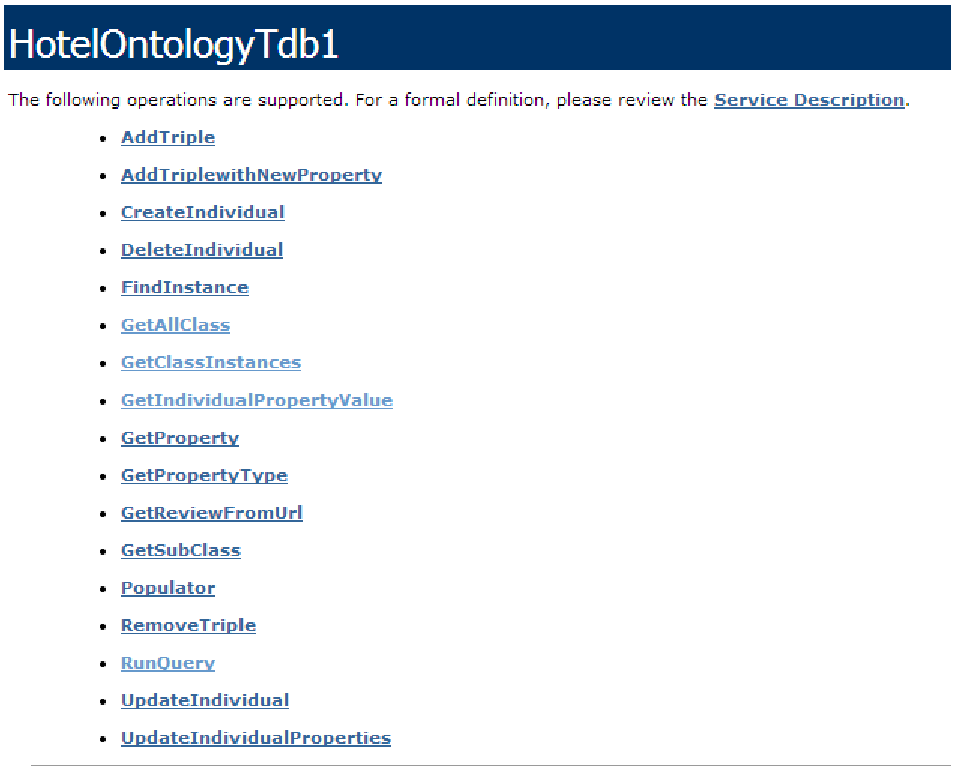
DeleteIndividual: Takes individual name and delete it from tdb.

GetAllClass: returns all hotel ontology model classes.

GetClassInstances: returns given class’s individuals as a list.

GetIndividualPropertyValue: returns individual with their properties’ values. Such as individual name is Hilton Hotel as parameter. It returns Hilton hotel stars 5, hotel theme seaside etc…

GetReviewFromUrl: This method returns hotel reviews as a string from given hotel url which both Booking.com and Tripadvisor.com. It using same html parsing methodologies.


HotelOntologyTdb1

The following operations are supported. For a formal definition, please review the Service Description.

- AddTriple
- AddTriplewithNewProperty
- CreateIndividual
- DeleteIndividual
- FindInstance
- GetAllClass
- GetClassInstances
- GetIndividualPropertyValue
- GetProperty
- GetPropertyType
- GetReviewFromUrl
- GetSubClass
- Populator
- RemoveTriple
- RunQuery
- UpdateIndividual
- UpdateIndividualProperties

Figure 5.9 Semantic Hotel Web Service Methods

5.4. Sentimental Analysis

Sentiment analysis is the process of detecting a piece of writing for positive, negative, or neutral feelings bound to it. Humans have the innate ability to determine sentiment; however, this process is time consuming, inconsistent, and costly in a business context. It’s just not realistic to have people individually read tens of thousands of user customer reviews and score them for sentiment. It is out of the scope of this thesis to develop sentimental analysis application. Therefore sentiment analysis tool “Sentigem” is used in this thesis.

Sentigem is a platform that offers an easy-to-use sentiment analysis tool for English language based documents or text blocks. To this end, we provide an API endpoint for computing sentiment from various consumption points determined by the user and passed to our API as text [53]

Sentigem api is used for analyzing reviews of hotels. The service is an English only service and The Sentigem API service only supports HTTPS calls. It is free api but needs the key. After sign up, the key is given. Sentigem takes key and text as a parameter then returns polarity of the text as positive, negative or neutral. Code snipped is shown in the figure 25.

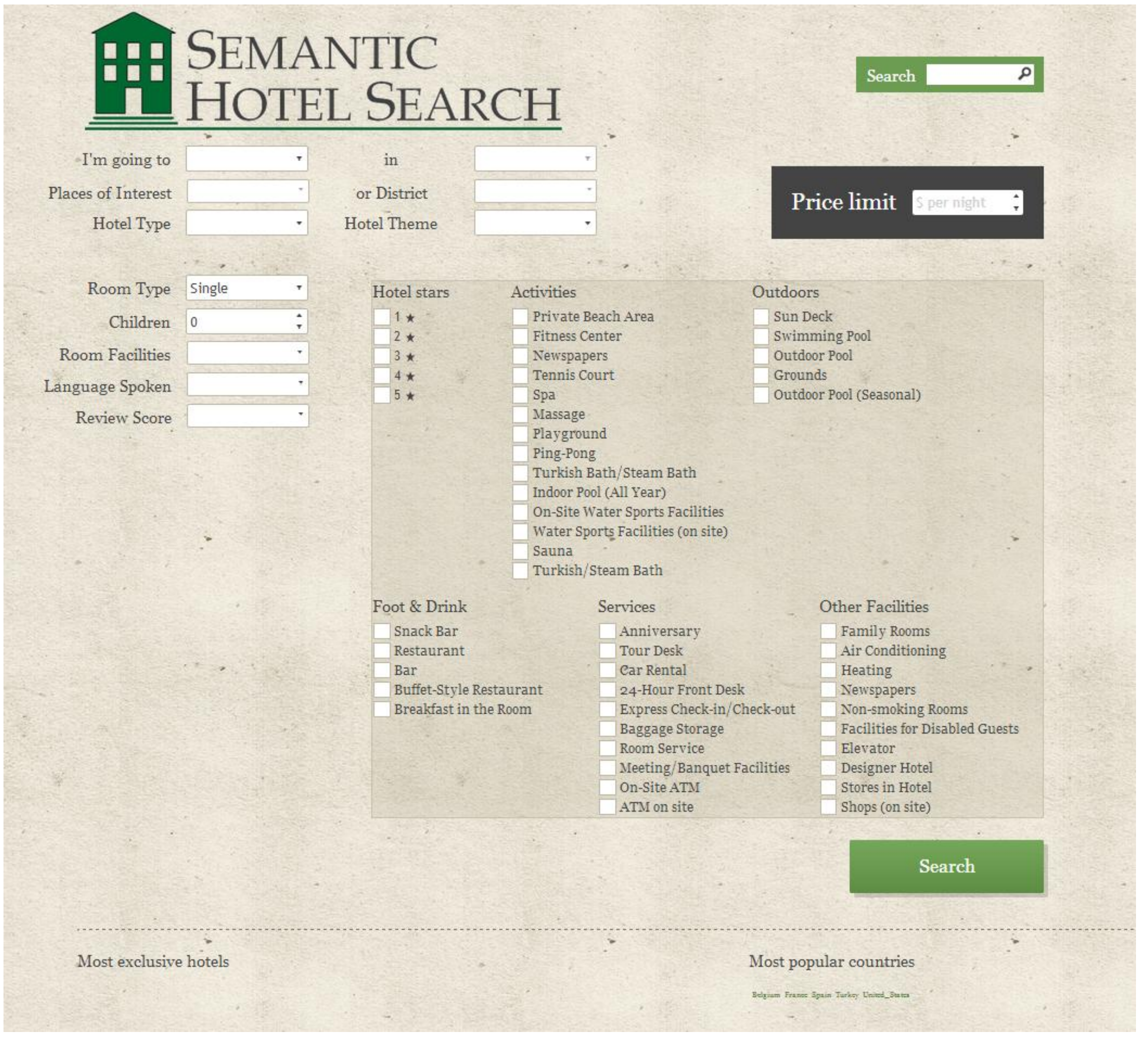
```
      WebClient client = new WebClient();
      string apikey =
"548102bd6d1d0c8edc1e32d484e71ca0t72sxgLNBuS_Yvb3zFCw9Pyr0j1TfVG
i";

      var uri = "https://api.sentigem.com/external/get-
sentiment?api-key=" + apikey + "&text=" + item;
      string text = client.DownloadString(uri);

              JavaScriptSerializer jss = new
JavaScriptSerializer();
              dynamic obj = jss.Deserialize<dynamic>(text);
              string type = obj["polarity"];
              DataRow row = dt.NewRow();
              row["Review"] = item;
              row["Situation"] = type;
              dt.Rows.Add(row);
```

5.5. Web Interface

The application, which is called Semantic Hotel Search is a .NET Web application providing a HTML user interface for the user, the query is presented to the system as illustrated in Figure 5.10.

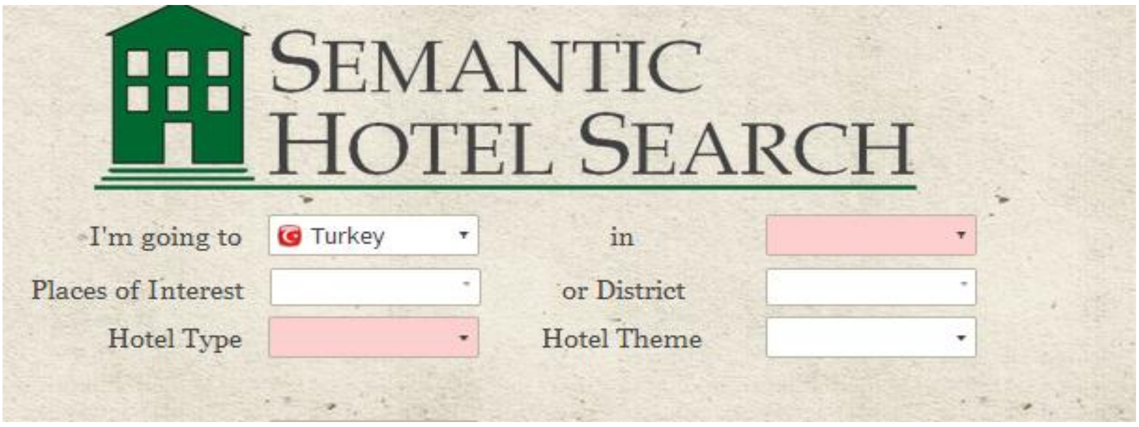


Room Type: Single
Children: 0

Figure 5.10 Semantic Hotel Search GUI

Once the user presses search, the query is processed by a Jena43 supported middleware Environment. Then the query has returned a list of matching results. Application interface is quite simple and user friendly. Users just needed to select field which they want to have features on the hotel. The users can create different requests to send to the server and processes the responses differently.

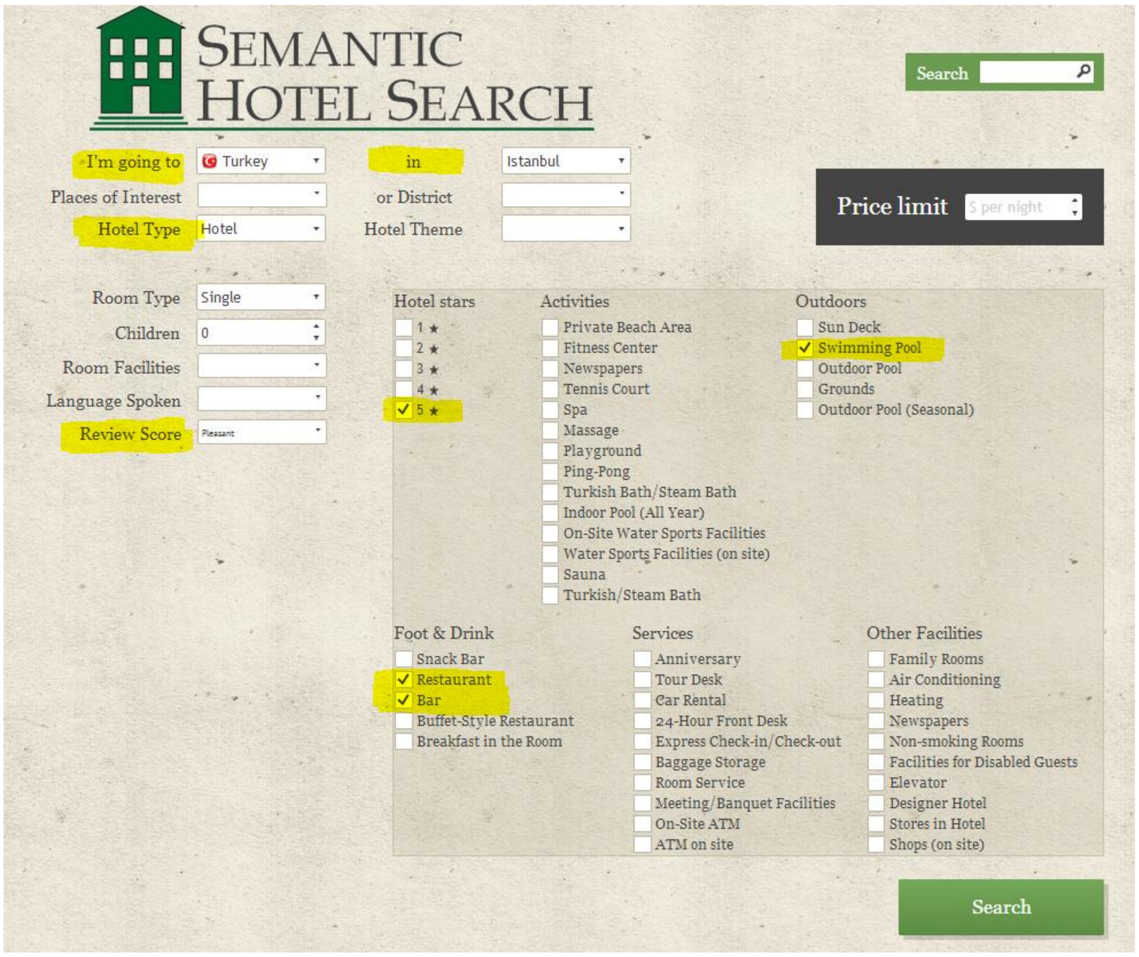
There are three required field to make query more significant. These are: Country, city and hotel type which they are shown in Figure 5.11.


SEMANTIC HOTEL SEARCH

I'm going to	Turkey	in	
Places of Interest		or District	
Hotel Type		Hotel Theme	

Figure 5.11 Required Fields of Application

As a given sample query; we are going to Istanbul, Turkey. We want five stars hotels which general review score is pleasant and more also have swimming pool, bar and restaurant. Then click search button.


SEMANTIC HOTEL SEARCH

Search

I'm going to	Turkey	in	Istanbul
Places of Interest		or District	
Hotel Type	Hotel	Hotel Theme	

Price limit $ per night

Room Type	Single
Children	0
Room Facilities	
Language Spoken	
Review Score	Pleasant

Hotel stars
- 1 ★
- 2 ★
- 3 ★
- 4 ★
- ✓ 5 ★

Activities
- Private Beach Area
- Fitness Center
- Newspapers
- Tennis Court
- Spa
- Massage
- Playground
- Ping-Pong
- Turkish Bath/Steam Bath
- Indoor Pool (All Year)
- On-Site Water Sports Facilities
- Water Sports Facilities (on site)
- Sauna
- Turkish/Steam Bath

Outdoors
- Sun Deck
- ✓ Swimming Pool
- Outdoor Pool
- Grounds
- Outdoor Pool (Seasonal)

Foot & Drink
- Snack Bar
- ✓ Restaurant
- ✓ Bar
- Buffet-Style Restaurant
- Breakfast in the Room

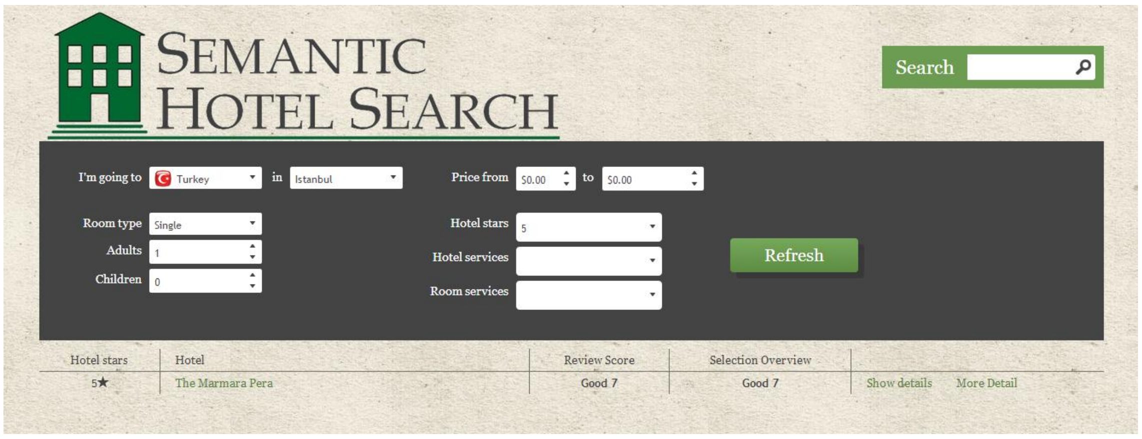
Services
- Anniversary
- Tour Desk
- Car Rental
- 24-Hour Front Desk
- Express Check-in/Check-out
- Baggage Storage
- Room Service
- Meeting/Banquet Facilities
- On-Site ATM
- ATM on site

Other Facilities
- Family Rooms
- Air Conditioning
- Heating
- Newspapers
- Non-smoking Rooms
- Facilities for Disabled Guests
- Elevator
- Designer Hotel
- Stores in Hotel
- Shops (on site)

Search

Figure 5.12 Sample Semantic Hotel Search

The query was constructed in the background of the application. Sample search result is shown in Figure 5.13



I'm going to Turkey in Istanbul

Price from $0.00 to $0.00

Room type Single

Adults 1

Children 0

Hotel stars 5

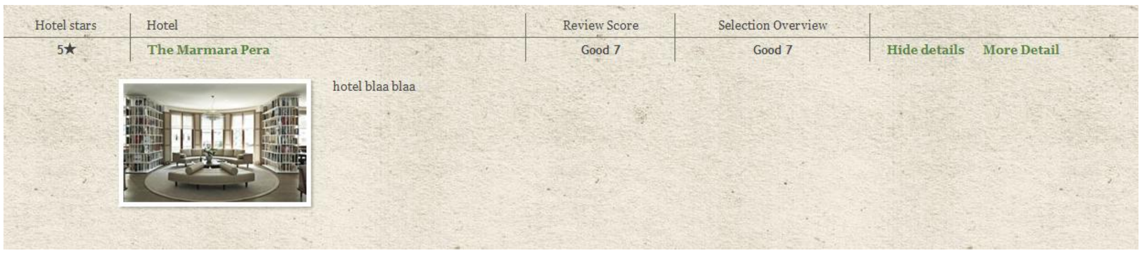
Hotel services

Room services

Hotel stars	Hotel	Review Score	Selection Overview	
5★	The Marmara Pera	Good 7	Good 7	Show details More Detail

Figure 5.13 Sample Search Result of Application

Users can see selected hotel features in short explanation as shown Figure 5.14.

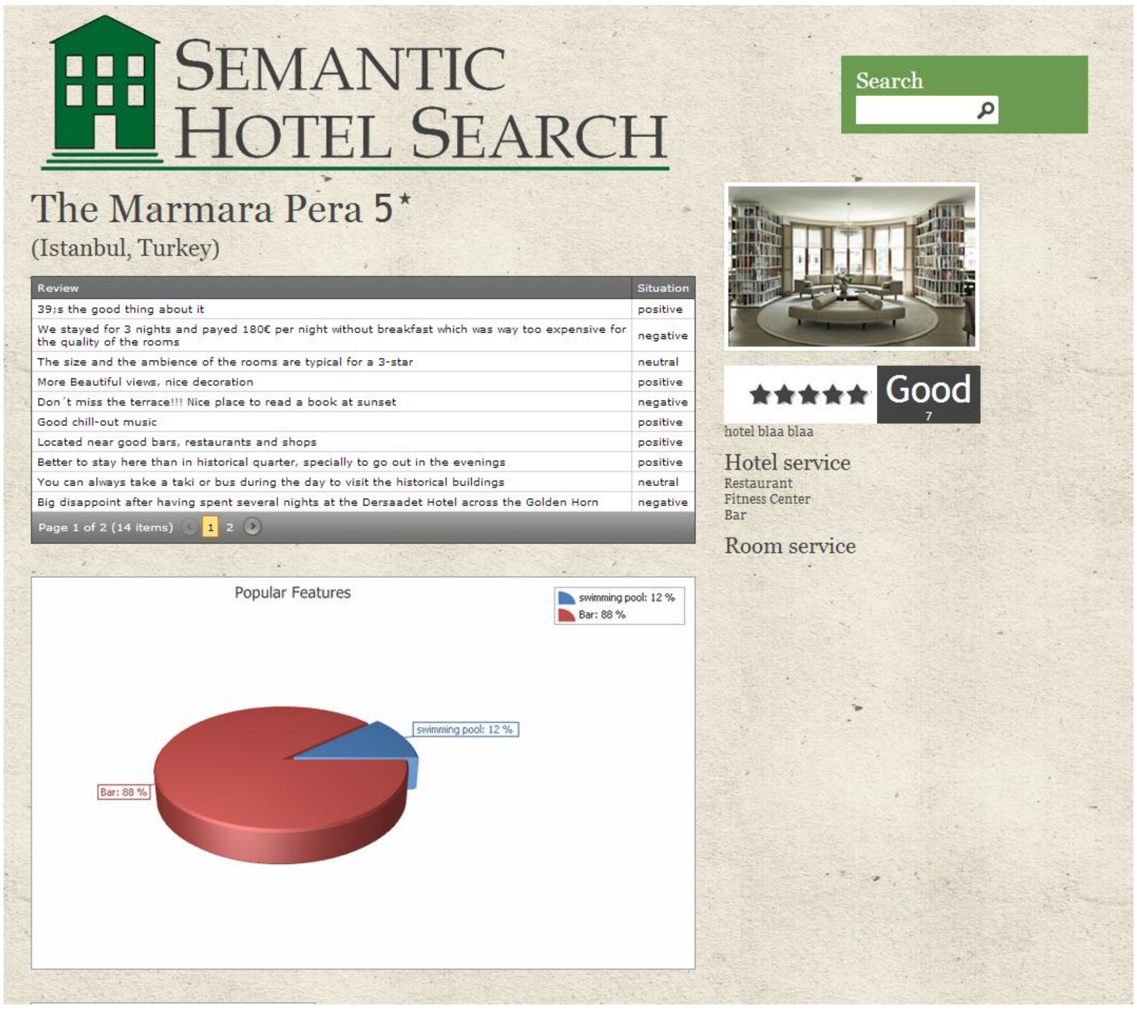


Hotel stars	Hotel	Review Score	Selection Overview	
5★	The Marmara Pera	Good 7	Good 7	Hide details More Detail

hotel blaa blaa

Figure 5.14 Selected Hotel Details of Application

If users want to see more details, they click more details and navigate to hotels page. In hotels own page, There are sentimental analysis about hotel and hotel facilities. There are review analysis table which show reviews and their polarities. Also there is graphic which show most popular features and most unfavoured features of the hotel.



SEMANTIC HOTEL SEARCH

Search

The Marmara Pera 5*

(Istanbul, Turkey)

Review	Situation
39;s the good thing about it	positive
We stayed for 3 nights and payed 180€ per night without breakfast which was way too expensive for the quality of the rooms	negative
The size and the ambience of the rooms are typical for a 3-star	neutral
More Beautiful views, nice decoration	positive
Don´t miss the terrace!!! Nice place to read a book at sunset	negative
Good chill-out music	positive
Located near good bars, restaurants and shops	positive
Better to stay here than in historical quarter, specially to go out in the evenings	positive
You can always take a taki or bus during the day to visit the historical buildings	neutral
Big disappoint after having spent several nights at the Dersaadet Hotel across the Golden Horn	negative

Page 1 of 2 (14 items) 1 2

Good 7

hotel blaa blaa

Hotel service

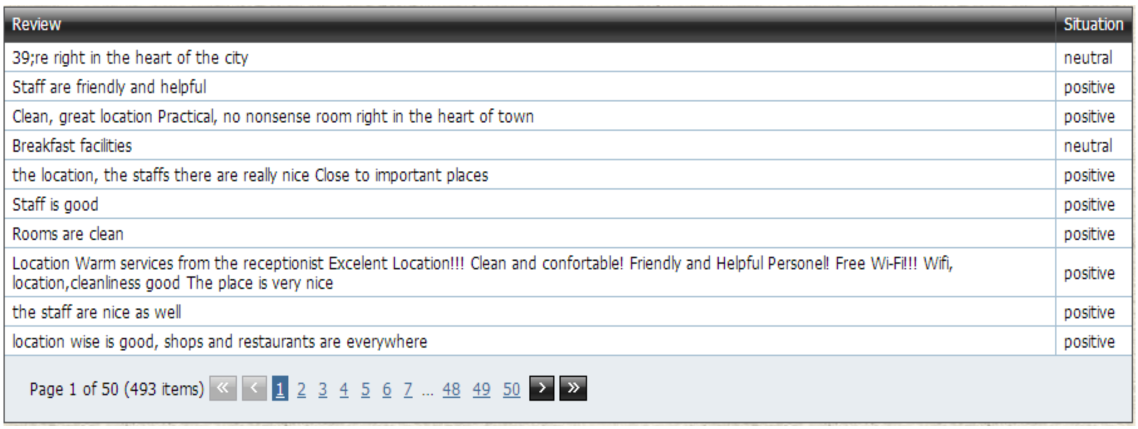
Restaurant
Fitness Center
Bar

Room service

Figure 5.15 The Selected Hotel Detail Page of Application

5.6. Evaluation

Test cases were run by using actual data from a set of about 400 accommodations located in five cities which of these are Barcelona, Madrid, Istanbul, Antalya, Muğla (Bodrum) in two countries; Turkey and Spain. 400 accommodations include about 250 Hotels, 60 hostels, 70 apartments, 20 bed and breakfast. For the test runs, several user profiles were supposed to be family, young couple, young friend groups, newly-wed couple etc… Each aggregating a list of different hotel preferences such as a price range, locational preferences, hotel facilities and so on. As observed from the Hotel Search project, Semantic web applications are efficient and easier to use who utilize them. Semantic hotel search provides more meaningful result, time efficiency and easiness in decision making. For example, in a normal hotel search system like tripadvisor.com, searching parameters are restricted. Only few parameters are selectable such as city, accommodation type and stars. Thus, according to these parameters, a wide range of hotels are returned. Some of them irrelevant, some of them less related with intended hotels and some of them are exactly expedient hotels. Therefore, analyzing each hotels, reading each reviews are highly time consuming process. Also, users need another search in different websites about hotel reviews. This means that same process must be repeated. For this reason it takes a lot of time to make decision about expedient hotel. On the other hand, searching parameters are various in semantic hotel search application. Hotel location, review types, stars, hotel indoor and outdoor activities, food, general specifications and services are selectable. Returned hotel list only consist of intended parameters. Each hotel has sentiment review analysis in their own page. Users can see whether facilities are good or not. Reviews are taken from two different big hotel search web sites (Tripadvisor.com and Booking.com) because of that, sentiment analysis is quite reliable.



Review	Situation
39;re right in the heart of the city	neutral
Staff are friendly and helpful	positive
Clean, great location Practical, no nonsense room right in the heart of town	positive
Breakfast facilities	neutral
the location, the staffs there are really nice Close to important places	positive
Staff is good	positive
Rooms are clean	positive
Location Warm services from the receptionist Excelent Location!!! Clean and confortable! Friendly and Helpful Personel! Free Wi-Fi!!! Wifi, location,cleanliness good The place is very nice	positive
the staff are nice as well	positive
location wise is good, shops and restaurants are everywhere	positive

Page 1 of 50 (493 items) « ‹ 1 2 3 4 5 6 7 ... 48 49 50 › »

Figure 5.16 Sample Hotel Review for Hotel Condal

	Positive	Negative	Neutral	Total	Accuracy
Human Analysis	325	73	95	493	100%
Semantic Hotel Search Analysis	263	57	173	493	84%

Table 5.2 Accuracy Rate Comparison of Sentimental Analysis

As shown on Table 5.2, application agree on 263 positive expressed sentiment out of 325, 57 negative sentiment out of 73 and sense more neutral sentiment, 173, than human analysis which is 95. There upon, it can be deduced from the table that application has difficulties to differentiate between positive and neutral sentiments. On the other hand if total accuracy is interpreted, %84 of cases are agreed on.

According to studies two humans will only agree on expressed sentiment in 82 percent of cases. Although human perception of sentiment seems to be more mature and accurate than an application's perception, it still is not 100% accurate. There have been various studies run by people and companies and they ended up with that average rate of human concordance is between 70% and 79%[15]. Thus, %84 accurate is satisfying for the conclusion. Consequently, semantic hotel search application provides quite easy way to users to provide making choice hotel that has desired in a very short time.

[15] Social Sentiment Sentiment Analysis [Internet]. *ON SOCIAL SENTIMENT AND SENTIMENT ANALYSIS*; http://brnrd.me/social-sentiment-sentiment-analysis/ (accessed date: 24 APR 2013)

CHAPTER 6

CONCLUSION

The main goal of this thesis and in the Semantic Hotel Search application is the integration of heterogeneous data sources and the (as far as possible) efficient and high performance application of Semantic Web technologies in the tourism domain. The current web provides data that is only loosely connected and cannot be used for rich querying. The model presented in this paper demonstrates the effectiveness of the semantic web for searching and extracting information. The semantic web, enabling meaningful search, provides us with a tool to extract fewer and more relevant data from the vast World Wide Web.

The Semantic Web element in this work is the knowledge base, and the contents of the knowledge base comprise not only the system data but also the data schema (i.e. metadata). The knowledge base is available through the Internet and therefore accessible in an OWL format by other systems and agents that will be able to understand and process its contents. It is also possible to augment this knowledge base with some complex description logics expressing additional constraints to the data, which is normally not possible when working with standard database systems.

The application has been designed in five independent parts: implementing the hotel domain ontology, Html parser, semantic services, sentimental analysis and Web interface to interact with the system. Initially we developed hotel domain ontology after researching and analyzing some hotel booking and tourism web sites. We take pilot websites www.booking.com and www.tripadvisor.com while creating class, subclass and their relations. Those descriptions and relation definitions could not have been implemented with usual relational database.

The architecture of the application is created in a way to overcome the difficulties normally faced in building such systems. One of the major challenges is to meaningfully integrate information in different web sites, as they do not have any standard or set criteria to express transportation facilities, hotel and accommodation, weather conditions or leisure activities. Therefore in order to integrate individuals, html parser is designed. Html parser takes URL as a parameter and convert webpages to

HtmlDocument type via HtmlAgilityPack. Then related information are extracted and individuals are saved to tdb using semantic web services.

Hotel Service application is a windows service project inside Semantic Hotel Search web application. This service runs every day at 00:00 automatically and update all hotel facilities, specialties and reviews. In consequence of, all hotel facilities, reviews, specifications are up to date and search result, sentimental analysis results are found accurately and consistently.

Semantic Hotel We Service was created in .Net platform and is bridge between web interface and windows service. It contains CRUD operations method, RunQuery method in order to run sparql queries, GetReviewfromUrl method to retrieve hotel review from Booking and Tripadvisor.

Sentimental analysis analyze hotel reviews and finds its polarities. For example, searching sentences includes swimming pool and returns general review polarity is positive, negative or neutral. What the most favorite features or scunner facilities are also searchable.

Semantic Hotel Search web interface makes it possible for the user to interact with the ontology processing unit in a human understandable way. The users can construct queries and search for hotels as they usual do in normal web sites. They do not have to construct RDF queries, instead the processing background of the application take criterion then converts it in actual RDF queries so that the user does not have to worry about how exactly the information is extracted.

The Semantic Web is the future of the Web. It is already being used by some industries such as the petroleum industry dealing with complex data sets. As different technologies and tools are being developed together with the experience being accumulated, Semantic Web will become the main technology used when dealing with complex data sets and information which are related to other concepts in a complex manner.

Not only Semantic web is growing technology, semantic web services are growing technology as much as semantic web. Enriching Web services with semantic information allows automatic location, composition, invocation, and interoperation of services. Significant work has already been done in this decade on Semantic Web services, and a large body of relevant work exists from earlier decades in fields such as knowledge representation, planning, agent-based systems, databases, programming languages, and software engineering. Nevertheless, many difficult research challenges remain, semantic web services are developing.

Several extensions and improvements can be added to the application. First of all, the information domain could have been described with richer properties and relations. For example, when creating a representational model for hotel, additional properties could have been included into the model such as the car rental. Also hotel price is now static and is not accurate. It will be dynamic and can change according to room type and time. Hotel booking system also can be included.

It is emphasized that the Semantic Web should not be viewed as a separate Web, but an extension of the current one, in which information and services are given well-defined meaning, thereby better enabling computers and people to work in cooperation. Dealing with heterogeneity has continued to be a key challenge since it was made possible to exchange and share data between computers and applications over the Internet. The tourism industry has been particularly affected by this heterogeneity because of its market fragmentation, and rather complex discovery and matchmaking tasks, including substitution and composition. Our experiences have shown that both travelers and travel service providers can benefit from the implementation of the Semantic Web-based scenario. Nevertheless, what we have learned is that there are some shortcomings to these new Internet technologies and emphasized, that if the Semantic Web is to be applied to the real-life scenarios the performance and expressiveness of Semantic Web technologies must be addressed.

REFERENCES

[1] Villarías, L.G. *Ontology-Based semantic querying of the web withrespect to food recipes*. MS thesis, Technical University of Denmark, 2004.

[2] Pranav Parikh, B.E. *Secured information integration with a semantic web-based framework*. MS thesis, The University of Texas at Dallas, 2009.

[3] Antoniou, G., and Frank van Harmelen. *A Semantic Web Primer*. Cambridge, Massachusetts: The MIT Press, 2004.

[4] Khosrowpour , M. *Encyclopedia of Information Science and Technology* Hershey, New York: Information Science Reference, 2005.

[5] Padilla, A., and Ashok Sahu. *A Report Submitted In Partial Fulfillment Of The Requirements For CS53.* http://www.armando.ws/wp content/uploads/2008/04/-web.doc, (accessed date: 21 Jan 2014).

[6] Liyang, Y., *Introduction to the Semantic Web and Semantic Web Services*. London: Springer-Verlag Berlin Heidelberg, 2011.

[7] Saleem, A., *Semantic Web Vision: survey of ontology mapping systems and evaluation of progress*. MS thesis, Blekinge Institute of Technology, 2006.

[8] W3[Internet]. RDF Notes; http://www.w3.org/DesignIssues/RDFnot.html, (accessed date: 12 Dec 2013).

[9] Institution of Information Science [Internet]. Papers; www.iis.sinica.edu.tw-/sw/SW1.ppt, (accessed 12 Dec 2014).

[10] Nédellec, C. and Nazarenko, A., *Ontologies and Information Extraction*. http://arxiv.org/ftp/cs/papers/0609/0609137.pdf. (accessed date: 21 Jan 2014).

[11] Shahzad, M., *A semantic web approach for dealing with university courses*. MS thesis, University of Adgar, 2013.

[12] Pioneer Of Journal[Internet], *Search Engine For Semantic Web*; http://-pioneerjournal.in/conferences/tech-knowledge/9th-national-conference/3514-search-engine-for-semantic-web.html, (accessed date: 21 Nov 2013).

[13] Decker, S. and Melnik, S., *The Semantic Web:The Roles of XML and RDF,* 2000, http://www.cs.vu.nl/~frankh/postscript/IEEE-IC00.pdf, (accessed date: 14 Dec 2013).

[14] RDF About[Internet], Home Page, http://www.rdfabout.com/intro/, (accessed date: 15 Dec 2013).

[15] Allemang, D. Hendle, J., *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Burlington, MA: Morgan Kaufmann Pub., 2011.

[16] Lemahieu, W., *Web service description, advertising and discovery: WSDL and beyond*,http://www.econ.kuleuven.be/public/NDBAA62/Downloadable%20papers/WSDLandBeyond.pdf, (accessed date: 23 Sep 2013).

[17] Baader, F., Horrocks, I. and Sattler, U., *Description Logics.* Elsevier, 2008.

[18] W3[Internet]. Web Ontology Language (OWL); http://www.w3c.org/TR/owl-features, (accessed date: 18 Dec 2013).

[19] Christiaan, M. and Klein, A., Change *Management for Distributed Ontologies*, Ph.D. diss., Vrije Universiteit, 2004.

[20] Mukhopadhyay, D., *A Model Approach to Build Basic Ontology*, http://arxiv.org/abs/1311.6245 (accessed date: 25 Nov 2013).

[21] Gruber, T. R., *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, [Journal] International Journal of Human-Computer Studies - Special issue: the role of formal ontology in the information technology archive Volume 43 Issue 5-6, Nov./Dec. 1995 Pages 907 - 928 .

[22] Computer Action Team[Internet], *What are OWL Ontologies?*, http://web.cecs.pdx.edu/~howe/cs410/owl_overview.pdf, (accessed date: 2 Jan 2014).

[23] Shahzad, M., *A semantic web approach for dealing with university courses*. MS thesis, University of Adgar, 2013.

[24] Juhnyoung, L., *Frequently Asked Questions on Ontology Technology*, IBM T.J. Watson Research Center, Hawthorne, NY, , 2004.

[25] Computer Action Team[Internet], *What are OWL Ontologies?*, http://web.cecs.pdx.edu/~howe/cs410/owl_overview.pdf, (accessed date: 2 Jan 2014).

[26] Informatics Review[Internet], *Applied Ontology*, http://www.informatics-review.com/wiki/index.php/Applied_ontology, (accessed date: 10 Jan 2014).

[27] Zhaohui, W. and Chen, H., *Semantic Grid: Model, Methodology, and Applications*, Hangzhou,; Hangzhou University Press, 2008.

[28] Rezaeiye, P.P., Fazli, M., Sharifzadeh, M., Moghaddam, H. and Gheisari, M., *Creating an ontology using protégé: concepts and taxonomies in brief*, http://www.wseas.us/e-library/conferences/2012/Sliema/MACMESE/MAC-MESE-56.pdf (accessed date : 23 Jan 2014).

[29] Jain, V. and Singh, M. *Ontology Development and Query Retrieval using Protégé Tool*, Published Online August 2013 in MECS (http://www.mecs-press.org/), I.J. Intelligent Systems and Applications, 2013, 09, 67-75 .

[30] Gali, A., Chen, C. X., Claypool, K. T. and Uceda-Sosa, R., *from ontology to relational databases*,http://www.cs.uml.edu/~cchen/ontology/comwim_04.pdf (accessed date: 10 Jan 2014).

[31] Marek Obitko, Jan Smid, Ontology Design with Formal Concept Analysis , Department of Cybernetics, Czech Technical University in Prague, Czech Republic.

[32] Henrik Bulskov Styltsvig, Ontology-based Information Retrieval, May 2006.

[33] Tar, H. and Nyunt, T.S., *Ontology-Based Concept Weighting for Text Documents* , International Conference on Information Communication and Management, 2011.

[34] Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, *Spinning The Semantic Web* , 2003.

[35] Juhnyoung, L., *Frequently Asked Questions on Ontology Technology*, IBM T. J. Watson Research Center, Hawthorne, NY, 2004.

[36] Denny,M.,OntologyToolsSurvey,Revisited, http://www.xml.com/pub/a/-2004/07/14/onto.html, (accessed date: 15 Feb 2014).

[37] Stanford University[Internet], Stanford University School of Medicine Resources, http://friendshost.com/university/stanford-university-school-of-medicine.php, (accessed date: 21 Jan 2014).

[38] XML[Internet], Ontologies, http://www.xml.com/pub/a/2002/11/06/-ontologies.html, Last access: 1.11.2004 (accessed date: 12 Jan 2014).

[39] Stanford University[Internet], http://www-ksl-svc.stanford.edu:5915/ (accessed date: 12 Jan 2014).

[40] Eldora[Internet], WebOnto, http://eldora.open.ac.uk:3000/webonto (accessed date : 12 Jan 2014).

[41] Archana, P. K., *Architecting and Designing of Semantic Web Based Application using the JENA and PROTÉGÉ -A Comprehensive Study*, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (3) , 2011, 1279-1282.

[42] Bioontology[Internet], Triple stores, http://www.bioontology.org/wiki/-images/6/6a/Triple_Stores.pdf (accessed date 13 Jan 2014).

[43] Zhang, Z., *Ontology Query Languages For The Semantic Web: A Performance Evaluation*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.3155-&rep=rep1&type=pdf (accessed date : 15 Dec 2013).

[44] Taniar, D. and Rahayu, R.W., *Web Semantics and Ontology*, ISBN 1-59140-907-1, 2006.

[45] W3[Internet]. Rdf Sparql Query; http://www.w3.org/TR/rdf-sparql-query/, (accessed date: 12 Dec 2013).

[46] Magnus Niemann, Malgorzata Mochol, Robert Tolksdorf, *Improving Online Hotel Search - What Do We Need Semantics For?* http://page.mi.fu-berlin.de/mochol/papers/semantics06.pdf (accessed date : 21 Jan 2014).

[47] Stanford University[Internet], Protege; http://protege.stanford.edu/aboutus-/aboutus.html (accessed date: 16 Jan 2014).

[48] Sentigem[Internet], About; http://sentigem.com/company/about (accessed date: 17 Jan 2014).

[49] Kim J.W.,*A Novel Approach to Ontology Management*, http://scholarworks.gsu.edu/cgi/viewcontent.cgi?article=1038&context=cis_diss 2010, (accessed date: 28 Dec 2013).

[50] Html Agility Pack; http://htmlagilitypack.codeplex.com/ (accessed date: 18 Jan 2014).

[51] Abrahams, B., *Tourism Information Systems Integration And Utilization Within The Semantic Web*, Ph.D. diss, Victoria University, 2006.

[52] Microsoft, Library; http://msdn.microsoft.com/en-us/library/ms950421.aspx (accessed date: 20 January 2014).

[53] Sentigem[Internet], About; http://sentigem.com/company/about (accessed date: 17 Jan 2014).

APPENDIX A

HOTEL DOMAIN ONTOLOGY

```
      <?xml version="1.0"?>
      <rdf:RDF

xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
         xmlns="http://www.owl-ontologies.com/hotel.owl#"
         xmlns:swrl="http://www.w3.org/2003/11/swrl#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
         xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"

xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl#"
         xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
       xml:base="http://www.owl-ontologies.com/hotel.owl">
       <owl:Ontology rdf:about="">
         <owl:versionInfo
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >1.0          by           Holger           Knublauch
(holger@smi.stanford.edu)</owl:versionInfo>
         <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >An      example      ontology      for      tutorial
purposes.</rdfs:comment>
         <owl:imports
rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"
/>
         <owl:imports
rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl"/>
       </owl:Ontology>
       <owl:Class rdf:ID="Hotel">
         <rdfs:subClassOf>
           <owl:Class rdf:ID="Accommodation"/>
         </rdfs:subClassOf>
       </owl:Class>
       <owl:Class rdf:ID="ReviewModel"/>
       <owl:Class rdf:ID="Facility"/>
       <owl:Class rdf:ID="PaymentMethod"/>
       <owl:Class rdf:ID="Outdoors">
         <rdfs:subClassOf rdf:resource="#Facility"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Destination"/>
<owl:Class rdf:ID="Museum">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="TouristAttraction"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TrainStation">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Transportation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Activities">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:Class rdf:ID="LanguageSpoken">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:Class rdf:ID="BusTerminal">
  <rdfs:subClassOf rdf:resource="#Transportation"/>
</owl:Class>
<owl:Class rdf:ID="Event"/>
<owl:Class rdf:ID="Service">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:Class rdf:ID="City">
  <rdfs:subClassOf rdf:resource="#Destination"/>
</owl:Class>
<owl:Class rdf:ID="General">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:Class rdf:ID="AirportHotel">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Accommodation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Comments"/>
<owl:Class rdf:ID="Food">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:Class rdf:ID="Concert">
  <rdfs:subClassOf rdf:resource="#Event"/>
</owl:Class>
<owl:Class rdf:ID="Room"/>
<owl:Class rdf:ID="HotelTheme"/>
<owl:Class rdf:ID="BudgetAccommodation">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Accommodation"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="hasStar"/>
          </owl:onProperty>
          <owl:someValuesFrom>
            <owl:Class>
              <owl:oneOf rdf:parseType="Collection">
```

```
                        <owl:NamedIndividual
rdf:ID="OneStarRating">
                          <rdf:type>
                            <owl:Class
rdf:ID="AccommodationRating"/>
                          </rdf:type>
                          <Star
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                          >1</Star>
                        </owl:NamedIndividual>
                        <owl:NamedIndividual
rdf:ID="TwoStarRating">
                          <rdf:type>
                            <owl:Class
rdf:about="#AccommodationRating"/>
                          </rdf:type>
                          <Star
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                          >2</Star>
                        </owl:NamedIndividual>
                      </owl:oneOf>
                    </owl:Class>
                  </owl:someValuesFrom>
                </owl:Restriction>
              </owl:intersectionOf>
            </owl:Class>
          </owl:equivalentClass>
          <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Accommodation  that  has  either  one  or  two  star
rating.</rdfs:comment>
        </owl:Class>
        <owl:Class rdf:about="#Accommodation">
          <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >A place to stay for tourists.</rdfs:comment>
        </owl:Class>
        <owl:Class rdf:ID="District">
          <rdfs:subClassOf rdf:resource="#Destination"/>
        </owl:Class>
        <owl:Class rdf:ID="Parks">
          <rdfs:subClassOf rdf:resource="#TouristAttraction"/>
        </owl:Class>
        <owl:Class rdf:about="#AccommodationRating">
          <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Consists of exactly three individuals.</rdfs:comment>
          <owl:equivalentClass>
            <owl:Class>
              <owl:oneOf rdf:parseType="Collection">
                <owl:NamedIndividual
rdf:about="#OneStarRating"/>
                <owl:NamedIndividual rdf:ID="FiveStarRating">
                  <Star
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                  >5</Star>
                  <rdf:type
rdf:resource="#AccommodationRating"/>
                </owl:NamedIndividual>
                <owl:NamedIndividual rdf:ID="FourStarRating">
                  <Star
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                  >4</Star>
                  <rdf:type
rdf:resource="#AccommodationRating"/>
                </owl:NamedIndividual>
                <owl:NamedIndividual
rdf:about="#TwoStarRating"/>
                <owl:NamedIndividual rdf:ID="ThreeStarRating">
                  <Star
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                  >3</Star>
                  <rdf:type
rdf:resource="#AccommodationRating"/>
                </owl:NamedIndividual>
              </owl:oneOf>
            </owl:Class>
          </owl:equivalentClass>
        </owl:Class>
        <owl:Class rdf:ID="RoomFacility">
          <rdfs:subClassOf rdf:resource="#Facility"/>
        </owl:Class>
        <owl:Class rdf:ID="Airport">
          <rdfs:subClassOf rdf:resource="#Transportation"/>
        </owl:Class>
        <owl:Class rdf:ID="StadiumArena">
          <rdfs:subClassOf rdf:resource="#TouristAttraction"/>
        </owl:Class>
        <owl:Class rdf:ID="Policies"/>
        <owl:Class rdf:ID="Internet">
          <rdfs:subClassOf rdf:resource="#Facility"/>
        </owl:Class>
        <owl:Class rdf:ID="Festival">
          <rdfs:subClassOf rdf:resource="#Event"/>
        </owl:Class>
        <owl:Class rdf:ID="Campground">
          <rdfs:subClassOf rdf:resource="#Accommodation"/>
          <rdfs:subClassOf>
            <owl:Restriction>
              <owl:onProperty>
                <owl:ObjectProperty rdf:about="#hasStar"/>
              </owl:onProperty>
              <owl:hasValue rdf:resource="#OneStarRating"/>
            </owl:Restriction>
          </rdfs:subClassOf>
          <owl:disjointWith rdf:resource="#Hotel"/>
        </owl:Class>
        <owl:Class rdf:ID="Monument">
          <rdfs:subClassOf rdf:resource="#TouristAttraction"/>
        </owl:Class>
        <owl:Class rdf:ID="Port">
          <rdfs:subClassOf rdf:resource="#Transportation"/>
        </owl:Class>
```

```
<owl:Class rdf:ID="Country">
  <rdfs:subClassOf rdf:resource="#Destination"/>
</owl:Class>
<owl:Class rdf:ID="ShoppingAreas">
  <rdfs:subClassOf rdf:resource="#TouristAttraction"/>
</owl:Class>
<owl:Class rdf:ID="Parking">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:Class rdf:ID="BedAndBreakfast">
  <owl:disjointWith rdf:resource="#Campground"/>
  <owl:disjointWith rdf:resource="#Hotel"/>
  <rdfs:subClassOf rdf:resource="#Accommodation"/>
</owl:Class>
<owl:Class rdf:ID="Resorts">
  <rdfs:subClassOf rdf:resource="#Accommodation"/>
</owl:Class>
<owl:Class rdf:ID="HotelHighlightValue"/>
<owl:Class rdf:ID="Highlight">
  <rdfs:subClassOf rdf:resource="#Facility"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#hasStar">
  <rdfs:domain rdf:resource="#Accommodation"/>
  <rdfs:range rdf:resource="#AccommodationRating"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasReviewModel">
  <rdfs:range rdf:resource="#ReviewModel"/>
  <rdfs:domain rdf:resource="#Accommodation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasComment">
  <rdfs:domain rdf:resource="#Accommodation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTransportation">
  <rdfs:range rdf:resource="#Transportation"/>
  <rdfs:domain rdf:resource="#Accommodation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasBadFacility">
  <rdfs:range rdf:resource="#HotelHighlightValue"/>
  <rdfs:domain rdf:resource="#ReviewModel"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRoom">
  <rdfs:range rdf:resource="#Room"/>
  <rdfs:domain rdf:resource="#Accommodation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasName">
  <rdfs:range rdf:resource="#Highlight"/>
  <rdfs:domain rdf:resource="#HotelHighlightValue"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPayment">
  <rdfs:range rdf:resource="#PaymentMethod"/>
  <rdfs:domain rdf:resource="#Policies"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEvent">
  <rdfs:range rdf:resource="#Event"/>
  <rdfs:domain rdf:resource="#City"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDestination">
```

```
          <owl:inverseOf>
            <owl:ObjectProperty rdf:ID="isDestinationOf"/>
          </owl:inverseOf>
          <rdfs:domain rdf:resource="#Accommodation"/>
          <rdfs:range rdf:resource="#District"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasService"/>
        <owl:ObjectProperty rdf:ID="belongsToCountry">
          <rdfs:domain rdf:resource="#City"/>
          <rdfs:range rdf:resource="#Country"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="belongsToCity">
          <rdfs:range rdf:resource="#City"/>
          <rdfs:domain rdf:resource="#District"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasOtelFacility">
          <rdfs:domain rdf:resource="#Accommodation"/>
          <rdfs:range rdf:resource="#Facility"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="isOfferedAt">
          <rdfs:domain rdf:resource="#Event"/>
          <rdfs:range rdf:resource="#Destination"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasAttraction">
          <rdfs:domain rdf:resource="#City"/>
          <rdfs:range rdf:resource="#TouristAttraction"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasRoomFacility">
          <rdfs:range rdf:resource="#RoomFacility"/>
          <rdfs:domain rdf:resource="#Room"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasGoodFacility">
          <rdfs:range rdf:resource="#HotelHighlightValue"/>
          <rdfs:domain rdf:resource="#ReviewModel"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasPart"/>
        <owl:ObjectProperty rdf:about="#isDestinationOf">
          <rdfs:domain rdf:resource="#Destination"/>
          <rdfs:range rdf:resource="#Accommodation"/>
          <owl:inverseOf rdf:resource="#hasDestination"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasPolicy">
          <rdfs:domain rdf:resource="#Accommodation"/>
          <rdfs:range rdf:resource="#Policies"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="hasTheme">
          <rdfs:range rdf:resource="#HotelTheme"/>
          <rdfs:domain rdf:resource="#Accommodation"/>
        </owl:ObjectProperty>
        <owl:DatatypeProperty rdf:ID="Address">
          <rdfs:domain rdf:resource="#Accommodation"/>
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="IsFree">
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
```

```
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Star">
          <rdfs:domain rdf:resource="#AccommodationRating"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Explanation">
          <rdfs:domain>
            <owl:Class>
              <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Parking"/>
                <owl:Class rdf:about="#Internet"/>
              </owl:unionOf>
            </owl:Class>
          </rdfs:domain>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="RoomSize">
          <rdfs:domain rdf:resource="#Room"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Checkout">
          <rdfs:domain rdf:resource="#Policies"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Score">
          <rdfs:domain rdf:resource="#Comments"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="WifiPlaces">
          <rdfs:range>
            <owl:DataRange>
              <owl:oneOf rdf:parseType="Resource">
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >All Area</rdf:first>
                  </rdf:rest>
                  <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >Public Area</rdf:first>
                </rdf:rest>
                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Hotel Rooms</rdf:first>
              </owl:oneOf>
            </owl:DataRange>
          </rdfs:range>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="AllowedPets">
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
          <rdfs:domain rdf:resource="#Policies"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="ImageName">
          <rdfs:domain rdf:resource="#Accommodation"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Review">
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
          <rdfs:domain rdf:resource="#ReviewModel"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Rate">
          <rdfs:domain rdf:resource="#HotelHighlightValue"/>
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Currency">
          <rdfs:range>
            <owl:DataRange>
              <owl:oneOf rdf:parseType="Resource">
                <rdf:rest rdf:parseType="Resource">
                  <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >EURO</rdf:first>
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >TL</rdf:first>
                    <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                  </rdf:rest>
                </rdf:rest>
                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >DOLAR</rdf:first>
              </owl:oneOf>
            </owl:DataRange>
          </rdfs:range>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Checkin">
          <rdfs:domain rdf:resource="#Policies"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Description">
          <rdfs:domain rdf:resource="#Accommodation"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="BedSize"/>
        <owl:DatatypeProperty rdf:ID="Price">
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="TripAdvisorUrl">
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
          <rdfs:domain rdf:resource="#Accommodation"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="BookingUrl">
          <rdfs:domain rdf:resource="#Accommodation"/>
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="ShortName">
          <rdfs:domain rdf:resource="#Country"/>
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Polarity">
```

```
          <rdfs:domain rdf:resource="#HotelHighlightValue"/>
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="Name">
          <rdfs:domain>
            <owl:Class>
              <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Accommodation"/>
                <owl:Class rdf:about="#Facility"/>
                <owl:Class rdf:about="#Service"/>
                <owl:Class rdf:about="#Room"/>
              </owl:unionOf>
            </owl:Class>
          </rdfs:domain>
          <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="price">
          <rdfs:domain rdf:resource="#Accommodation"/>
        </owl:DatatypeProperty>
        <owl:NamedIndividual           rdf:about="http://www.owl-
ontologies.com/travel.owl#Cannes"/>
        <owl:NamedIndividual rdf:ID="BrÃ¼ksel">
          <belongsToCountry>
            <Country rdf:ID="Belgium">
              <ShortName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >be</ShortName>
            </Country>
          </belongsToCountry>
          <rdf:type rdf:resource="#City"/>
        </owl:NamedIndividual>
        <owl:NamedIndividual rdf:ID="Sagrada_Familia">
          <belongsToCity>
            <City rdf:ID="Barcelona">
              <hasAttraction>
                <StadiumArena rdf:ID="Camp_Nou"/>
              </hasAttraction>
              <hasAttraction>
                <Museum rdf:ID="Cosmo_Casia"/>
              </hasAttraction>
              <belongsToCountry>
                <Country rdf:ID="Spain">
                  <ShortName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >es</ShortName>
                </Country>
              </belongsToCountry>
            </City>
          </belongsToCity>
          <rdf:type rdf:resource="#District"/>
        </owl:NamedIndividual>
        <PaymentMethod rdf:ID="Paypal"/>
        <RoomFacility rdf:ID="Bathroom"/>
        <LanguageSpoken rdf:ID="Turkish"/>
        <HotelTheme rdf:ID="Beach_SeaSide"/>
        <City rdf:ID="MuÄŸla">
          <belongsToCountry>
            <Country rdf:ID="Turkey">
              <ShortName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >tr</ShortName>
            </Country>
          </belongsToCountry>
        </City>
        <HotelTheme rdf:ID="Luxury"/>
        <City rdf:ID="Amsterdam"/>
        <City rdf:ID="Paris">
          <belongsToCountry>
            <Country rdf:ID="France">
              <ShortName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >fr</ShortName>
            </Country>
          </belongsToCountry>
        </City>
        <HotelTheme rdf:ID="Budget_Backpacker"/>
        <Comments rdf:ID="Wonderful">
          <Score
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >9</Score>
        </Comments>
        <District rdf:ID="Chamartin">
          <belongsToCity>
            <owl:NamedIndividual rdf:ID="Madrid">
              <rdf:type rdf:resource="#City"/>
              <belongsToCountry rdf:resource="#Spain"/>
            </owl:NamedIndividual>
          </belongsToCity>
        </District>
        <owl:NamedIndividual rdf:ID="AydÄ±n">
          <rdf:type rdf:resource="#City"/>
          <belongsToCountry rdf:resource="#Turkey"/>
        </owl:NamedIndividual>
        <HotelTheme rdf:ID="Gourmet"/>
        <Resorts rdf:ID="Radisson_Blu_Resort_Spa_Cesme">
          <BookingUrl
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.booking.com/hotel/tr/radisson-blu-resort-
spa-cesme.en-
us.html?sid=8aa80ffe4b7682747d7b6adc8a479b24;dcid=1#hash-
blockdisplay4</BookingUrl>
          <TripAdvisorUrl
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.tripadvisor.com/Hotel_Review-g298004-
d1229186-Reviews-Radisson_Blu_Resort_Spa_Cesme-
Cesme_Izmir_Province_Turkish_Aegean_Coast.html</TripAdvisorUrl>
          <Name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Radisson Blu Resort &amp; Spa Cesme</Name>
          <hasReviewModel>
            <ReviewModel
rdf:ID="Radisson_Blu_Resort_Spa_Cesme_ReviewModel"/>
```

```
          </hasReviewModel>
        </Resorts>
        <District rdf:ID="Besiktas">
          <belongsToCity>
            <City rdf:ID="Istanbul">
              <hasAttraction rdf:resource="#Cosmo_Casia"/>
              <hasAttraction>
                <Monument rdf:ID="Columbus_Monument"/>
              </hasAttraction>
              <belongsToCountry rdf:resource="#Turkey"/>
            </City>
          </belongsToCity>
          <hasAttraction>
            <Monument rdf:ID="Dolmabahce_SarayÄ±"/>
          </hasAttraction>
        </District>
        <Airport rdf:ID="Sabiha_Gokcen_Airport"/>
        <Highlight rdf:ID="Highlight_Bar">
          <Name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Bar</Name>
          <Polarity
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >positive</Polarity>
          <Rate
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >10</Rate>
        </Highlight>
        <Comments rdf:ID="Exceptional">
          <Score
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >9.5</Score>
        </Comments>
        <owl:NamedIndividual           rdf:about="http://www.owl-
ontologies.com/travel.owl#Bursa"/>
        <RoomFacility rdf:ID="Flat-Screen_Tv"/>
        <RoomFacility rdf:ID="Safe"/>
        <City rdf:ID="Sydney"/>
        <City rdf:ID="Miami_Beach">
          <belongsToCountry>
            <Country rdf:ID="United_States">
              <ShortName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >us</ShortName>
            </Country>
          </belongsToCountry>
        </City>
        <owl:AllDifferent>
          <owl:distinctMembers rdf:parseType="Collection">
            <Hotel rdf:ID="The_Marmara_Pera">
              <Address
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Mesrutiyet    Cad.,Tepebasi,    Beyoglu,    34430
Istanbul</Address>
              <hasComment>
                <Comments rdf:ID="Good">
```

```
                  <Score
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >7</Score>
                </Comments>
              </hasComment>
              <hasReviewModel>
                <ReviewModel
rdf:ID="The_Marmara_Pera_ReviewModel"/>
              </hasReviewModel>
              <Name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >The Marmara Pera</Name>
              <hasOtelFacility>
                <Food rdf:ID="Restaurant"/>
              </hasOtelFacility>
              <hasStar rdf:resource="#FiveStarRating"/>
              <hasService>
                <Service rdf:ID="Anniversary"/>
              </hasService>
              <hasPolicy>
                <Policies rdf:ID="TheMarmaraPera_Policies">
                  <AllowedPets
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
                  >false</AllowedPets>
                  <Checkin
rdf:datatype="http://www.w3.org/2001/XMLSchema#time"
                  >14:00:00</Checkin>
                  <Checkout
rdf:datatype="http://www.w3.org/2001/XMLSchema#time"
                  >12:00:00</Checkout>
                  <hasPayment>
                    <PaymentMethod rdf:ID="Visa"/>
                  </hasPayment>
                  <hasPayment>
                    <PaymentMethod rdf:ID="MasterCard"/>
                  </hasPayment>
                  <hasPayment>
                    <PaymentMethod rdf:ID="AmericanExpress"/>
                  </hasPayment>
                  <hasPayment>
                    <PaymentMethod rdf:ID="Cash"/>
                  </hasPayment>
                </Policies>
              </hasPolicy>
              <hasDestination>
                <District rdf:ID="Taksim">
                  <isDestinationOf
rdf:resource="#The_Marmara_Pera"/>
                  <belongsToCity rdf:resource="#Istanbul"/>
                  <hasAttraction>
                    <Parks rdf:ID="Gezi_ParkÄ±"/>
                  </hasAttraction>
                </District>
              </hasDestination>
              <BookingUrl
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
              >http://www.booking.com/hotel/tr/the-marmara-
pera.en-
gb.html?sid=4a1a6b7ff2d2eb6670533713f56de5da;dcid=1#hash-
blockdisplay1</BookingUrl>
              <ImageName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >http://r-
ec.bstatic.com/images/hotel/max300/189/18904632.jpg</ImageName>
              <hasTheme>
                <HotelTheme rdf:ID="City_Tour"/>
              </hasTheme>
              <TripAdvisorUrl
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >http://www.tripadvisor.com/Hotel_Review-g293974-
d530256-Reviews-The_Marmara_Pera_Hotel-
Istanbul.html</TripAdvisorUrl>
              <hasTransportation>
                <Airport
rdf:ID="Ataturk_International_Airport"/>
              </hasTransportation>
              <hasRoom>
                <Room rdf:ID="Standard_Double_or_Twin_Room">
                  <RoomSize
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >2</RoomSize>
                  <hasRoomFacility>
                    <RoomFacility rdf:ID="Heating"/>
                  </hasRoomFacility>
                </Room>
              </hasRoom>
              <hasOtelFacility>
                <Activities rdf:ID="Fitness_Center"/>
              </hasOtelFacility>
              <hasOtelFacility>
                <Food rdf:ID="Bar"/>
              </hasOtelFacility>
              <Description
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >hotel blaa blaa</Description>
            </Hotel>
          </owl:distinctMembers>
        </owl:AllDifferent>
        <Food rdf:ID="Snack_Bar"/>
        <HotelTheme rdf:ID="Family"/>
        <HotelTheme rdf:ID="Natural_Wildlife"/>
        <owl:AllDifferent>
          <owl:distinctMembers rdf:parseType="Collection">
            <owl:NamedIndividual rdf:about="#OneStarRating"/>
            <owl:NamedIndividual rdf:about="#ThreeStarRating"/>
          </owl:distinctMembers>
        </owl:AllDifferent>
        <Comments rdf:ID="Pleasant">
          <Score
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >6</Score>
        </Comments>
        <owl:AllDifferent>
```

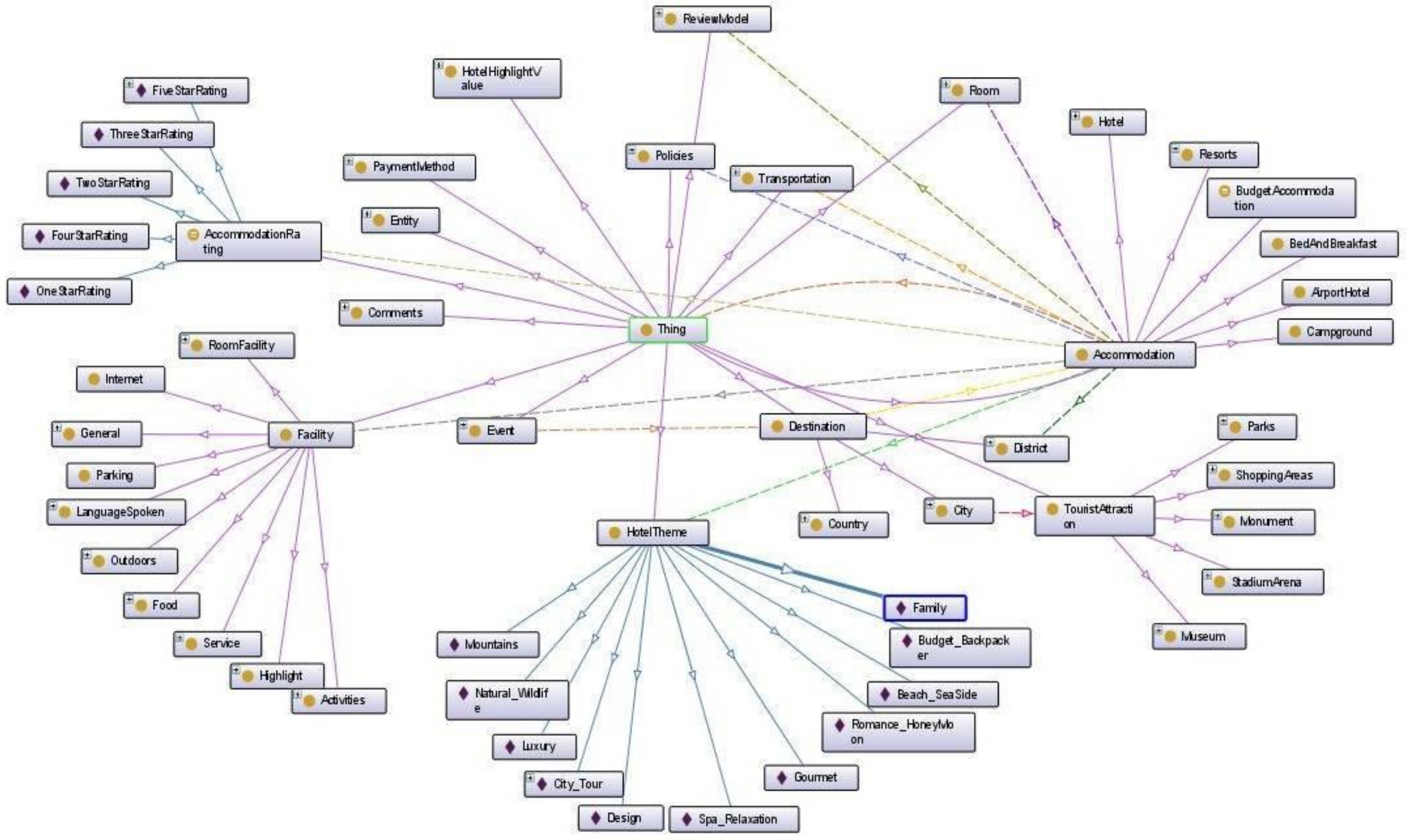
```
          <owl:distinctMembers rdf:parseType="Collection">
            <owl:NamedIndividual rdf:about="#ThreeStarRating"/>
            <owl:NamedIndividual rdf:about="#TwoStarRating"/>
          </owl:distinctMembers>
        </owl:AllDifferent>
        <owl:AllDifferent/>
        <owl:NamedIndividual rdf:ID="Barcelonata">
          <belongsToCity rdf:resource="#Barcelona"/>
          <rdf:type rdf:resource="#District"/>
        </owl:NamedIndividual>
        <Activities rdf:ID="Private_Beach_Area"/>
        <Highlight rdf:ID="Highlight_Swimming_pool">
          <Name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >swimming pool</Name>
        </Highlight>
        <Comments rdf:ID="Very_Good">
          <Score
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >8</Score>
        </Comments>
        <Outdoors rdf:ID="Sun_Deck"/>
        <HotelTheme rdf:ID="Romance_HoneyMoon"/>
        <General rdf:ID="Family_Rooms"/>
        <RoomFacility rdf:ID="Iron"/>
        <owl:NamedIndividual rdf:ID="Downtown_Barcelona">
          <rdf:type rdf:resource="#District"/>
          <belongsToCity rdf:resource="#Barcelona"/>
        </owl:NamedIndividual>
        <City rdf:ID="Antalya">
          <belongsToCountry rdf:resource="#Turkey"/>
        </City>
        <HotelTheme rdf:ID="Mountains"/>
        <owl:NamedIndividual rdf:ID="Cairns">
          <rdf:type rdf:resource="#City"/>
        </owl:NamedIndividual>
        <HotelHighlightValue rdf:ID="HotelHighlightValue_1"/>
        <Parks rdf:ID="Park_Guell"/>
        <LanguageSpoken rdf:ID="Spanish"/>
        <LanguageSpoken rdf:ID="French"/>
        <District rdf:ID="KadÄ±koy">
          <belongsToCity rdf:resource="#Istanbul"/>
          <hasAttraction>
            <ShoppingAreas rdf:ID="Optimum"/>
          </hasAttraction>
        </District>
        <owl:NamedIndividual rdf:ID="Raval">
          <rdf:type rdf:resource="#District"/>
          <belongsToCity rdf:resource="#Barcelona"/>
        </owl:NamedIndividual>
        <District rdf:ID="Moda">
          <belongsToCity rdf:resource="#Istanbul"/>
        </District>
        <HotelTheme rdf:ID="Design"/>
        <General rdf:ID="Air_Conditioning"/>
        <ShoppingAreas
rdf:ID="The_Centre_Comercial_MaremÃ gnumis"/>
```

```
      <owl:NamedIndividual rdf:ID="Ankara">
        <rdf:type rdf:resource="#City"/>
        <belongsToCountry rdf:resource="#Turkey"/>
      </owl:NamedIndividual>
      <owl:AllDifferent>
        <owl:distinctMembers rdf:parseType="Collection">
          <owl:NamedIndividual rdf:about="#OneStarRating"/>
          <owl:NamedIndividual rdf:about="#TwoStarRating"/>
        </owl:distinctMembers>
      </owl:AllDifferent>
      <HotelTheme rdf:ID="Spa_Relaxation"/>
      <City rdf:ID="Nice">
        <belongsToCountry rdf:resource="#France"/>
      </City>
      <Airport rdf:ID="Adnan_Menderes_Airport"/>
      <District rdf:ID="Salamanca">
        <belongsToCity rdf:resource="#Madrid"/>
      </District>
      <Outdoors rdf:ID="Swimming_Pool"/>
      <owl:NamedIndividual rdf:ID="Ramblas">
        <rdf:type rdf:resource="#District"/>
        <belongsToCity rdf:resource="#Barcelona"/>
      </owl:NamedIndividual>
      <owl:NamedIndividual rdf:ID="El_Born">
        <rdf:type rdf:resource="#District"/>
      </owl:NamedIndividual>
      <RoomFacility rdf:ID="Toilet"/>
      <Service rdf:ID="Tour_Desk"/>
      <Service rdf:ID="Car_Rental"/>
      <City rdf:ID="Izmir">
        <belongsToCountry rdf:resource="#Turkey"/>
      </City>
      <City rdf:ID="Ibiza">
        <belongsToCountry rdf:resource="#Spain"/>
      </City>
      <Outdoors rdf:ID="Outdoor_Pool"/>
      <LanguageSpoken rdf:ID="English"/>
      <District rdf:ID="Madrid_Center">
        <belongsToCity rdf:resource="#Madrid"/>
      </District>
      <Comments rdf:ID="Excellent">
        <Score
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >8.5</Score>
      </Comments>
    </rdf:RDF>

    <!-- Created with Protege (with OWL Plugin 3.5, Build 649)
http://protege.stanford.edu -->
```

APPENDIX B

HOTEL DOMAIN ONTOLOGY ONTO GRAF VISUALIZATION


ReviewModel
HotelHighlightValue
FiveStarRating
ThreeStarRating
TwoStarRating
FourStarRating
AccommodationRating
OneStarRating
PaymentMethod
Entity
Comments
Policies
Transportation
Room
Hotel
Resorts
BudgetAccommodation
BedAndBreakfast
AirportHotel
Campground
Accommodation
Thing
RoomFacility
Internet
General
Facility
Event
Destination
District
Parking
LanguageSpoken
Outdoors
Food
Service
Highlight
Activities
HotelTheme
Country
City
TouristAttraction
Parks
ShoppingAreas
Monument
StadiumArena
Museum
Family
Budget_Backpacker
Beach_SeaSide
Romance_HoneyMoon
Gourmet
Mountains
Natural_Wildlife
Luxury
City_Tour
Design
Spa_Relaxation