2010-01-01

Semantic Geospatial Search and Ranking in the Context of the Geographical Information System TerraFly

Mortadha Ali Alkhawaja
University of Miami, m.alkhawaja@umiami.edu

UNIVERSITY OF MIAMI

SEMANTIC GEOSPATIAL SEARCH AND RANKING IN THE CONTEXT OF THE GEOGRAPHICAL INFORMATION SYSTEM TERRAFLY

By

Mortadha Alkhawaja

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Computer Science

Coral Gables, Florida

August 2010

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of

the requirements for the degree of

Master of Computer Science

SEMANTIC GEOSPATIAL SEARCH AND RANKING IN THE CONTEXT OF THE GEOGRAPHICAL INFORMATION SYSTEM TERRAFLY

Mortadha Alkhawaja

Approved:

_____
Ubbo Visser, Ph.D.
Professor of Computer Science

_____
Terri A. Scandura, Ph.D.
Dean of the Graduate School

_____
Geoff Sutcliffe, Ph.D.
Professor of Computer Science

_____
Henryk Marcinkiewicz, Ph.D.
Professor of Instructional Systems
Aramco Services Company
Houston, Texas

MORTADHA ALKHAWAJA (M.S., Computer Science)

Semantic Geospatial Search And Ranking in the Context of The Geographical Information System TerraFly (August 2010)

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Ubbo Visser.
No. of pages in text. (140)

Modern Web based GIS systems have responded significantly to semantic Web technology as it offers opportunities to overcome interoperability and integration problems. There are abundant needs especially for the systems intending to provide more than just a map with basic geographical information. More sophisticated systems can offer more than navigation services and can integrate with several data sources, thereby providing a richer, wider and highly usable information service to be used in business, governmental and different life domains. Search is an essential part of any GIS system because of the huge amount of data representing different meanings that are stored in one or distributed data sources. A model is presented which focuses on searching for geospatial information to answer query semantics rather than query syntax. This model used the most recent and approved standards among the semantic Web communities, and was applied on TerraFly a GIS system. Since ranking is a critical factor in measuring the quality of any search engine, a ranking algorithm is also proposed and evaluated.

ACKNOWLEDGEMENTS

I would like to thank Professor Ubbo Visser for his valuable time in guiding me, answering my questions and providing the different kinds of advice and support needed to complete this thesis. I would also like to thank my committee members, Professor Geoff Sutcliffe and Professor Henryk Marcinkiewicz, for their guidance and support. I am greatly thankful to my wife, Fatimah, for her patience and support during my studies and to my daughter, Hawraa, and my son, Danial, for understanding how busy I was. Special thanks to my parents for their efforts in educating and encouraging me; without their efforts I would not have been able to achieve this.

Finally, I would like to thank my advisor and the director of the graduate studies in the Department of Computer Science, Professor Dilip Sarkar, for providing the different kinds of advice and support needed during my Master's study at the University of Miami.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Web 1.0 was limited in sharing knowledge and published resources. It provided the ability to transform printed contents to published online resource, creating personal Web pages and advertizing for business organizations [Guh09]. Web 2.0, which is the current trend, is more advanced in social networking, conducting business, information sharing and search. For example, Web 2.0 enabled the revolution of internet businesses like Amazon and e-Bay [Guh09]. People search the internet for almost everything on a daily basis. In addition, advanced technology enables audio and video conferencing through the Web with sometimes less cost than using cell phones. Knowledge sharing is an essential part of Web 2.0 [Guh09] and benefits from the rapid development of technology like network speed, storage capability, parallel processing, etc. There is a wider range of usage of the Web in almost all life domains including commercial, governmental and personal domains.

With Web 2.0 we are now able to browse, retrieve, download, share, save and perform many operations on the huge amount of information on the Web. One storage drive is now able to store tens if not hundreds of thousands of books. E-mail boxes contain huge amounts of data including their attachments. Moreover, information on the Web is stored and retrieved as HTML, text-based, multimedia, and as many other formats, which are presentable by computers and understandable by humans. However, Berners-Lee has proposed Web 3.0, a new Web through which machines can understand and extract information from the data available and routed on the Web [Guh09]. Web 3.0 also known as Semantic Web [Guh09] is a promise to store, process and treat data differently for the purpose of more intelligent systems and better data integration.

Thousands of messages in your e-mail inbox will no more be only tractable by you trying to confirm a fact. Machines should be able to do that with our help. That is with our preparation started from agreeing on a uniform language, uniform data storage, uniform query language and building on all these standards. Moreover, the theory of sharing only documents will change to become sharing documents, knowledge, data and linking them to construct a global Web [BHBL09]. In this work, we focus on applying some of these standards for the purpose of searching for semantic geospatial data on knowledge bases available for use on the Web and integrating this data with TerraFly [RCC+03], a GIS (Geographical Information System).

1.1 Geospatial Semantic Web:

There is a belief that 80% of all information has a geographical component. It includes maps, environmental information, planning, and information from other domains that references address [ST07].

What makes geographical data important with regard to the semantic Web is the variety of their models, formats, semantics and relations. Moreover, in order to employ the Web as a medium for data and information integration, comprehensive datasets and vocabularies are required as they enable the disambiguation and alignment of other data and information [ALH09]. All these factors are considered challenges in the design and implementation of a semantic Web based GIS system. Since meanings of data can be interpreted differently, users searching for geo-locations can express their questions differently with varied intensions. Ontologies can be used as a shared agreement between domain members in order to describe the knowledge of that domain.

Geographical information is easy for humans and hard for computers to understand because of its complex structures and relationships. If computers were able to interpret geo-information on the Web they could substitute them with icons drawn on the map. The Semantic Web has been designed to enable machines to understand geo-data for interoperability and integration among available geospatial data sources. Human understanding and interpretation of the data are not easy tasks to be applied on machines as they are not able to reason the same way humans do. Even if they could, supplying all the required rules for them to be able to interpret things like humans may not yet be in our grasp due to the human brain complexity. Ontology reasoners were created in order to extract implicit knowledge and use available information to infer more hidden facts. This is especially true where it is unlikely that geo and topological relations between all the different objects in a knowledge base are explicitly defined.

Place ontologies are essential parts of geographical search engines as they support the effective retrieval of the referenced resources [ASJ07]. Many of the geographic relations such as "*nearTo*" or "*southOf*" are only defined at the ontology level between concepts and can be easily discovered through human visualization. However, there is no evidence of their existence as they do not exist at the instance level. Therefore, it is a requirement to develop extractors that can compute such relationships geometrically.

TerraFly, a public service of Florida International University sponsored by NSF (MII, MRI et al.) and by NASA, USGS, and IBM is our candidate on which we would like to apply semantic Web techniques for the purpose of integrating data from various knowledge bases, and enriching

its search service with more accurate and meaningful query results [Uni10a]. TerraFly was selected for some reasons like its need to integrate with different applications. Moreover, the current formats of TerraFly lack a mechanism for semantic integration with other richer knowledge bases. The nature of the planned integrated data is semantic-based and comes from different available public knowledge bases such as DBPedia [ABK+08] and Geonames [Geo10].

DBPedia is our main alternative data source in which we search for geo-data to add a richer layer to TerraFly. DBPedia is an open and public data repository resulted from extracting information from Wikipedia [Wik10] infoboxes and articles, then publishing them in RDF (Resource Description Framework) triples as shown in figure 1-1 [ABK+08]. Moreover, DBpedia plays the role of a central interlinking hub, this means if we are able to connect to DBpedia, then we are also connected to data sources such as Geonames, the World Factbook, UMBEL, EuroStat, YAGO [ALH09] and more.

Integration with DBPedia was accomplished through the development of a Semantic Geo Search Service (SGSS) through which users can locate geo-points on earth by searching for places. SGSS uses various available standards, tools and APIs to accomplish the task. It uses WordNet [Uni10b] which is as a lexical database to redefine terminologies. OWL, RDF and SPARQL describe, store and query the semantic data. Protégé builds and maintains ontologies and Pellet reasons over the geo-ontology. Jena passes the SPARQL query to DBPedia.

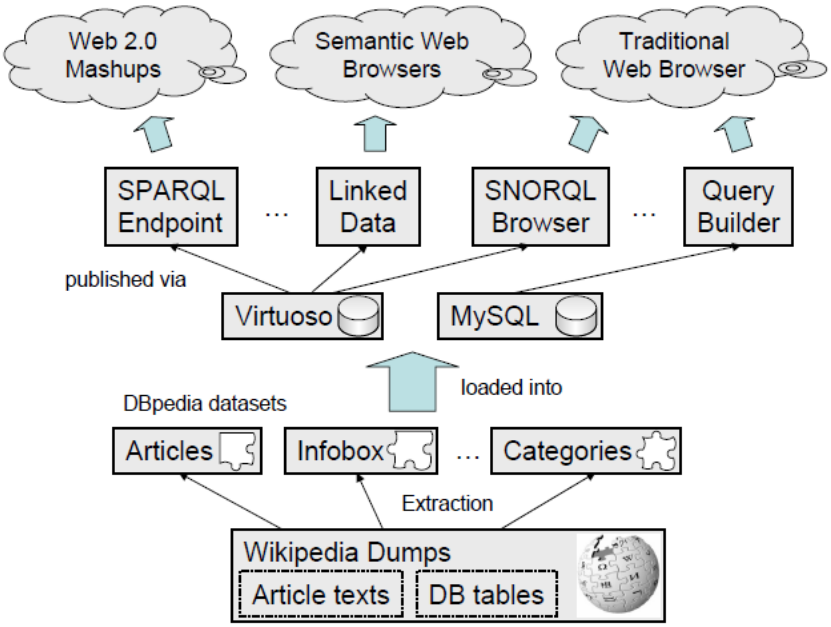

Figure 1-1: DBPedia datasets which are extracted from Wikipedia knowledge base and stored into Virtuoso and MySQL databases. On top of that, there are publication interfaces through which other knowledge bases connect to the databases and pass queries. Source: [ABK+08]

As the user query in SGSS is treated semantically, adding another dimension (the semantic dimension) should increase the number of results but many times reduces accuracy. This is due to the fact that it searches for all potential meanings of the query whereas only one or two of them really reflect the user intention. Therefore, retrieving all results that meet all potential meanings requires a ranking mechanism to order the results retrieved. A major focus in this research was the definition of a ranking mechanism that is different from traditional search ranking in several aspects. Our ranking is based on several factors. One factor indicates how well the retrieved document meets the meaning of the user query. Another indicates whether the retrieved document adheres to the concept of the place specified by the user. Finally, a factor evaluates whether the retrieved document is described as “under coverage” of the geospatial area

specified by the user and interpreted by the system. Discussion of these factors follows in this work.

Finally, there is an analysis of the evaluation experiments. It is based on user feedback. A general traditional search service was developed in order to compare SGSS algorithm with a keyword-based search algorithm and ranking. The traditional service uses the TF-IDF [SM86] algorithm to order the returned results. In the evaluation, we can show that SGSS surpasses the keyword-based search in terms of accuracy and recall. The SGSS produces a better accuracy if it is assigned different scoring weights. Improved weightings can still be achieved by conducting more experiments with a larger user group. At the same time, the traditional search engine was found faster than SGSS due to the extra operations involved such as gazetteer lookup, ontology processing, reasoning and the complexity of the semantic query union

Chapter 2. Background Concepts

1.1 **Search Engine:**

A search engine is a computer program developed to search one or several databases for one or several documents given user specifications in the form of a query. The resulting document can be any piece of information, a Web page, a file, a URL or even a collection of information, which is gathered and represented in a useful and meaningful manner.

The search process goes through several steps. In general, the following steps are used:

1- Interpreting user query.
2- Translating the interpreted query to its equivalent database or knowledge base query.
3- Running the query on one or several data or knowledge sources and retrieving a set of documents.
4- Filtering to remove irrelevant results. This step is usually performed during the query processing.
5- Ranking the remaining relevant documents.

However, most keyword-based search engines like Yahoo, Google and MSN do not spend time on the first step to really interpret the meanings of the user query which the semantic search engines do [TSB09].

2.2 Semantic Search Engine:

The difference between a semantic search engine and a conventional search engine is that the former is meaning-based [TSB09]. Since the perfect search engine is the one that finds the most precise documents in order to meet the expectations of the user, several semantic based approaches have been developed to accomplish this task. This is true especially with the rapid development of the semantic Web tools, standards, data models, query languages and reasoners. The available technologies and tools enhance the Web data retrieval in many ways. For example, many traditional or keyword-based search engines crawl the contents of the databases, match the words of the user query, and are using a ranking mechanism to rank the results containing the query words or some of them. Later, it is the user's responsibility to navigate from one result page to the other and to choose his/her closest result from the lists. In contrast, some researchers consider the semantic search engines to be aware of meaning as these engines are proposed to give a special attention to the real meaning intended by the user.

Ranking can be defined as the ability to return relevant documents first [BCC+08]. It is generally assumed that each search engine assigns a score to each result that satisfies a Boolean search criteria and then sorts the results according to this score [BCCW05]. Search algorithms use factors to evaluate each result and incorporate all the factor scores into one in order to achieve the ranking. The main differences between keyword based ranking algorithms and semantic ranking algorithms are summarized as follows. While the semantic search ranks according to the presence and quality of metadata, keyword-based systems like Lucene (http://lucene.apache.org/solr) enable ranking of documents according to :

(1) Their ability to match the keyword-based query.

(2) The keywords used in anchor links (i.e. the text associated to hyperlinks pointing to a specific document).

(3) The document popularity measured as function of the weight of the links referring to the document itself [BCC+08].

2.3 Semantic Web:

The semantic Web is defined as an extension of the current Web, in which information is given well-defined meaning, enabling computers and people to work together [BLHL01]. The key factor for machine processibility of content on the semantic Web is that data should be self-describing which can be achieved by producing a common language to define both the data and its metadata [KBM08]. The semantic Web helps computers to interpret and infer implicit knowledge and information by providing a more structured, linked and interpretable data formats e.g the RDF. Moreover, it uses languages like XML based RDF, RDF Schema and OWL (Web Ontology Language) by which data can be defined, described, verified and linked to other data with some relationships if that exist. Each of the previously mentioned language has its own purpose and each one is built on the basis of another as illustrated in the semantic Web layer in the following section. Recently, commercial database companies like ORACLE have incorporated functions into their DBMS for the purpose of creating, storing and retrieving billions of RDF triples [BCT07] that contributes to the overall success of the semantic Web.

The semantic Web layers illustrated in the figure 2-1 give an overview of the semantic Web design and ultimate goal. The URI layer defines the basis on which we build our models. The data and language models must be based on international character sets. Moreover, the URI layer means every object must have a unique identifier. XML defines a standard to write structured

and hierarchical data documents and can be described by XML schema. RDF is an XML based language to describe resources and RDFS is meant for laying the RDF resources, properties, and relationships into hierarchical structures in addition to defining basic ontologies. However, ontology vocabulary is needed to describe more complex ontology structures and relationships. Currently, different OWL languages have been proposed like OWL-Full OWL-DL and OWL-Lite by the W3C Web Ontology Working Group [W3C04]. Each is considered an extension of RDFS providing more features and restrictions with the benefit of efficient reasoning and expressivity. The logic layer is to extend the ontology by defining rules and the proof layer is to execute these rules. Trust is evaluating whether the proof is enough to trust the operations and quality of information provided by the Web. Finally, the semantic applications are built on top of the trust layer and therefore we can develop them based on the quality of the available information and operations [AvH08].

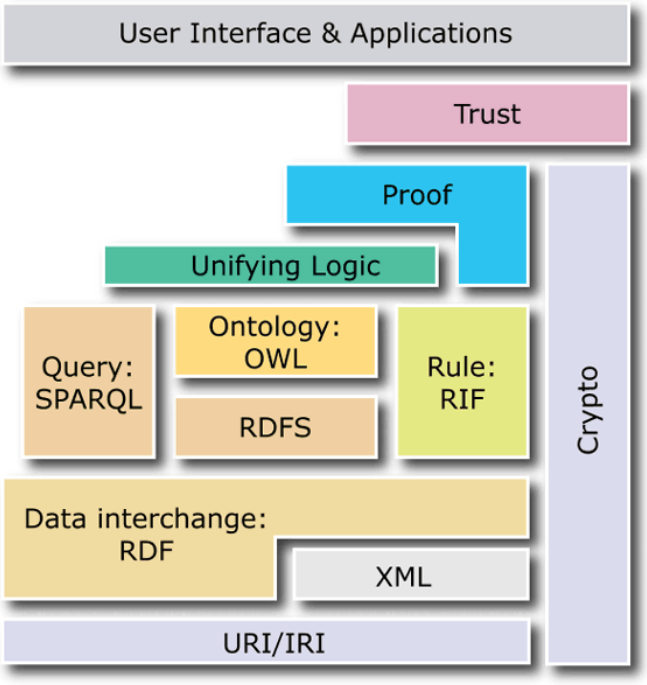

Figure 2-1: The semantic Web layers. Source: [SEM10]

2.4 Ontology:

Ontology is "an explicit specification of a conceptualization" as defined by [Gru93]. The standard and most typical ontologies are known to be theories that use a specific vocabulary to describe entities, classes, properties, and functions related to a certain view of the world [FEAC02]. Ontology describes concepts, relationships and other objects in a structured format in order to enable further processing and reasoning tasks in future. Ontology also offers *formal semantics* [AvH08] which describes the meaning of knowledge accurately as being applied in mathematical logic. Such semantics enable reasoning, checking consistency between ontology and knowledge, performing instance classification and to redefine meanings of terms for different programming purposes.

2.4.1 Ontology Classification:

Although the ontology is offering a full and comprehensive description of concepts and relationships, different ontologies have different expressive powers and complexities. Breitman [BCT07] classified ontologies according to their level of expressiveness as follows:

- **Controlled vocabularies:** describes a limited and specific number of concepts that are related to a specific field or application. It is widely used in classification systems like products classification systems used in industry.
- **Glossaries:** describes terms with their meaning in a natural language.
- **Thesauri:** describes and defines lexical concepts and relates them to each other in a hierarchical structure.
- **Informal is-a hierarchy:** describes hierarchical structure between concepts. The relations between the classes are not necessary adhering to the is-a relationship but it was

designed for a specific categorization system to be used by some applications. For example, accommodation and tour might appear under tourism although they do not hold the is-a relationship.

- **Formal is-a hierarchy:** describes hierarchical concepts that adhere the is-a relationship.
- **Frames:** describes classes and their properties or attributes.
- **Ontology with value restriction:** apply very limited functionality of the intended use of the ontology. It defines a range of values for the instances or properties of the ontology concepts.
- **Ontology with logical constructs:** an ontology that describes concepts, properties and relationships and is able to add logical rules in first order logic.

Definition: "A directory ontology is defined as an ontology containing only concepts and hypernym/hyponym relationships between them" [ZLT+08]. Defining such hierarchical relationships is known as "*subClassOf*" in OWL. The simplicity of these ontologies makes them easy to be integrated to different applications and therefore they were integrated in the directory ontology to the main DBPedia ontology for the purpose of filling the gap and missing information in the later one.

There is another classification of ontologies based on their generality like upper level and domain ontologies. An upper level ontology describes general concepts of the world that are not specific to an application or domain. In contrast, a domain ontology describes the shared and agreed conceptualization of knowledge in a specific field such as geography by defining the taxonomy and rules to be applied about different concepts of the domain [GHM+08]. OWL

ontologies have been developed and used in many areas like e-science, medicine, biology, geography, astronomy, defense, and the automotive and aerospace industries [GHM+08].

The main purpose of the domain ontology is to generate instances, link them, validate new instances and infer implicit instances that were not explicitly defined. Moreover, missing information from an instance can be discovered by validating the instance against the domain ontology. As a result, mandatory relationships and properties that are defined by a class in the domain ontology but are not discovered to be part of the instance can be reported as an error in that instance for further correction. However, creating the domain ontology requires an agreement between domain experts in order to develop a unified understanding for the goal of future data and knowledge integration. Therefore, creating a perfect and comprehensive ontology for a domain is usually a difficult task, a reason why we are integrating multiple ontologies into our main geospatial ontology. "The benefits of creating the ontologies will outweigh the cost and efforts in developing them" [FEAC02].

2.4.2 Ontology Use:

Ontology represents a vocabulary and everyone can create a vocabulary using well known standards like OWL, RDF and RDF schema. One benefit of using these standards is to be able to map different vocabularies published by different applications or organizations. RDF triples help achieving such mapping. An RDF triple consists of *Subject-predicate-Object*. While the subject and object both refer to an existing resource on the Web identified by a URI, the predicate describes the relation between them and is also identified by a URI. Our use of the ontology in this research is to redefine geographical concepts, find their general and specialized forms and find implicit concepts and relations between them. All of these uses are for the goal of enriching

and reformulating user query in order to enhance the search algorithm. Other fields in which ontologies are being used include:

- Natural language processing
- Extracting knowledge from text
- Improving data retrieval
- Solving the heterogeneity problem between different data sources.
- Describing Web services in an unambiguous and computer interpretable format.

2.4.3 Ontology Languages:

There are three languages for OWL (Ontology Web Language) namely OWL-Full, OWL-DL, and OWL-Lite. OWL Full is a complete language without limitations. It uses RDF and RDF schema and is able to change the meanings of the predefined primitives of RDF or OWL. OWL Full is fully compatible with RDF and RDFS but its expressive power leads to inefficient or even undecidable conclusion. For a particular task, logic is decidable if it is possible to design an algorithm that will terminate in a finite number of steps [Pro03].

OWL-DL stands for OWL description logic and is more efficient than OWL-Full but at the expense of full compatibility with RDF and RDFS. Finally, OWL-Lite is even considered the most efficient language particularly suited for reasoning tasks but it has less expressive power than the previous ones [AvH08]. Some work has been done to improve the first generation of OWL languages as currently there are some issues in the complexity of their syntax. For

example, because OWL uses RDF triples, many facts are represented in OWL but they also require adding more objects to the construct [GHM+08].

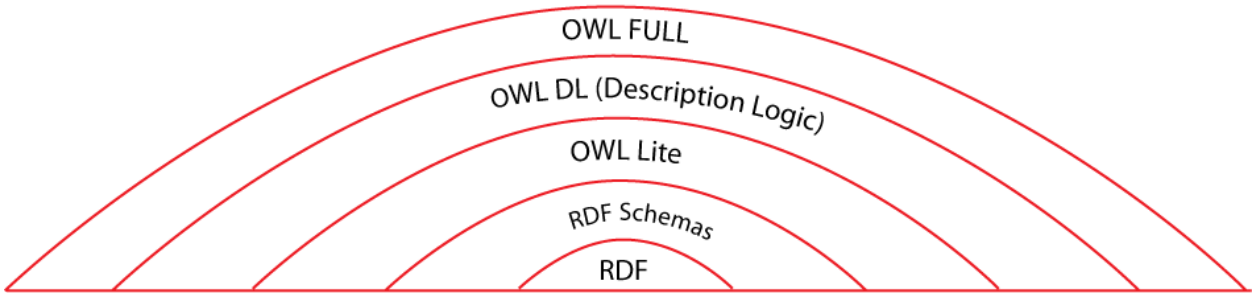

Figure 2-2: OWL languages and their subset structure. Source: http://www.w3.org

2.4.4 Class:

A class represents a grouping of objects with similar properties or features. The term *Class* has been used in several contexts in computer science to define a hierarchy of related concepts inheriting some properties from their parents. The class concept in semantic Web defines an abstract model that can be instantiated to yield one or more objects or instances. One of the benefits in developing a taxonomy is defining a generalization or a specialization of a concept through which the application is able to provide semi alternatives for that class. Moreover, when a subclass is defined, it must hold the "Is A" relationship with its super class and therefore that subclass inherits all the properties of its super class. OWL classes can relate to their equivalent classes with *owl:equivelentClass* relationship. This relation explicitly states that two classes are equivalent where sometimes it is necessary to discover implicit equivalency using inferences.

2.4.5 Instance:

An instance —also known as individual— is a concrete object of its class. An example can be "*UM is a University*" so the *UM* is the instance of the abstract class *University* because *UM* satisfied the object properties of *University* such as *hasStudents*. However, perfect decision whether an instance belongs to a specific class or not is sometimes difficult to achieve especially if the source of that instance is not perfect. For example, an instance of a class *University* is not verifiable because it satisfies all the properties of the class *School* but does not indicate if it offers degrees or not.

2.4.6 Object Property:

The object property is a binary relation between the set of instances of two classes. In its simplest form, it defines a relationship between an object and another in the underlying ontology. Each relationship can have several characteristics such as symmetry, transitivity, asymmetry, reflexivity ... etc. An example can be "*UM isLocatedIn Miami*" in which *UM* is an object, *isLocatedIn* is a geographical relationship that connects the first object with another namely *Miami*. Reasoning over object properties is also possible. For example, knowing that, "*Miami is located southOf Orlando*", where the definition of "*southOf*" is marked as an inverse relation of "*northOf*", can infer the fact that "*Orlando is located northOf Miami*".

2.4.7 Data Type Property:

Data type property defines a binary relationship between the set of individuals represented by a class and another set of instances of a datatype [BCT07]. In other words, the data type property defines the domain restrictions and range values that must be followed by different data objects that belong to a specific data type. Examples of data type properties are integer, double or date just to mention some. Figure 2-3 expresses a simple structure of an ontology.

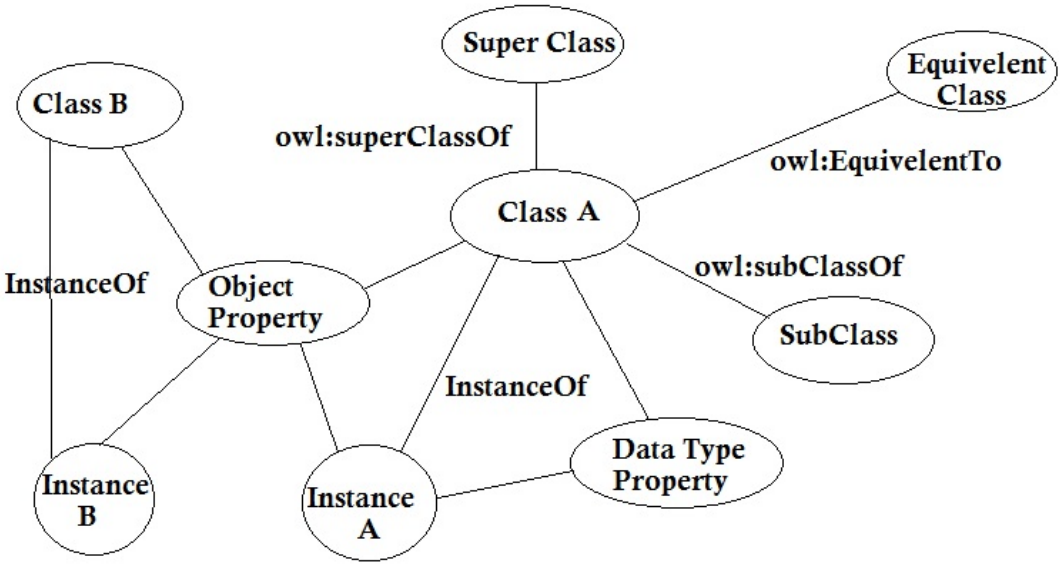

Figure 2-3: A subset of the general ontology structure.

2.5 First Order Logic:

Logic is the study of reasoning [Wel05]. It is used to draw conclusions from a set of logical statements known as assertions. Assertions are listed in an ordered way that constitutes an argument and they are called the "premises" for that argument. First order logic is considered the

foundation in knowledge representation and has many roles with regards to knowledge representation [AvH08]:

- It has a high expressive power through a high-level language.
- It describes logical statements through formal semantics, which is very important for disambiguation of meanings.
- It defines precise consequences from logic statements.
- A proof system exists. The proof system enables us to trace back the details that have lead to a consequence.
- A proof system establishes meaning that the derived statements are obtained semantically from the premises. More expressive logics like HOL (Higher Order Logic) do not have such proof systems.

2.6 Description Logic:

Description logic is a set of knowledge representation formalisms that describe concepts with their objects and properties using rules [BCT07]. It is a subset of predicate logic that has an efficient proof system. Concepts and rules are related through what is called “axioms” from which it derives its reasoning. Such structure and featuring makes it particularly important for ontological reasoning in the semantic Web. Breitman [BCT07] mentioned that the reasoning techniques should at least cover:

- Concept Derivation: The ability to reason over the concept class-subclass relationships.
- Concept Instantiation: The ability to reason over the class-instance relationships.

2.7 Reasoner:

A semantic reasoner is a computer program able to infer logical consequences from a set of asserted facts and axioms [KOD09]. Inferring properties and relationships between concepts; and concluding additional facts from instances and class descriptions is referred to as Ontological Reasoning. The Racer reasoner is an example, which checks a concept's consistency and reclassification using an inference mechanism that is done within the framework of Description Logics [LA04]. Description logic is a subset of predicate logic (First Order Logic) to perform efficient reasoning. Although most of the ontology-reasoning systems are based on description logic, some reasoners use rules and first order logic [KBM08].

2.8 SPARQL:

SPARQL is the standard query language for querying RDF documents and was approved by the W3C in 2008 [PS08]. It can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via a middleware [PS08]. Since the nature of RDF data is based on graph patterns, SPARQL can retrieve a set of results or RDF graphs. A SPARQL query processor will try to find a subgraph that best meets the query basic graph pattern which consists of *Subject-Predicate-Object.*

2.9 Geoparsing:

Geotagging is the process of assigning geospatial context information [ST07] to, for example, assign geo-coordinates or to define geometric boundaries. Geoparsing is the process of

recognizing and searching for the information that has been processed by geotagging. Information extractors play an important role in geotagging as it is their responsibility in extracting geo-coordinates and location information and annotate different information resources. In this research, DBPedia was analyzed and an algorithm was applied to geoparse the geographical resources.

2.10 Gazetteers:

Gazeetteers are databases of geospatial features stored with their names, locations, descriptions and some numerical geo information. Gazetteers are widely used by GIS systems as feed points that supply data to areas of interest over maps. There are several gazetteers available on the Web, like Geonames, the US Geological Survey's Geospatial Names Information System (GNIS) and ADL gazetteer. Initially, Geonames was to be used, but there were performance issues due to limited service with regard to a license. Therefore, an original gazetteer was created which was extracted from GNIS (http://geonames.usgs.gov/). The current Gazetteer covers locations within the US.

Chapter 3. TerraFly

3.1 Overview:

TerraFly is a Web-enabled geographical information system that allows users to virtually fly over remotely sensed data, including satellite imagery and aerial photography, using a standard Internet browser [RCC+03]. It retrieves data from multiple databases and correlates address fields in each database to layer objects of interest over a map. Moreover, TerraFly uses US census data, government and public databases for environment, economy and healthcare in order to add a richer layer of information to each application in the system. Several applications such as real estate marketing, urban planning and environmental monitoring are to be used in TerraFly with each application using multiple data sources with different formats and carrying different meanings. Moreover, the variety of the geospatial data models, formats, semantics and relations makes the integration process more complicated than integrating other simpler data. Consequently, this kind of heterogeneous data use by one system makes TerraFly a good candidate to apply the semantic Web techniques for the benefit of integrating multiple data layers from several Web data sources and to simplify the process as the amount of data grows with time.

Currently, TerraFly is widely used for B2B services like real estate, and aspires to develop and integrate more data sources as demand increases. Although some other applications like urban planning, wind & solar energy planning have been thought of, they are still under study because

of the heterogeneous data sources that make the integration difficult. The TerraFly group is dreaming of a world class service that can produce a portal point opening the gate to as much information available on the Web as possible. The semantic Web technology was identified as one component necessary to achieve this dream. In figure 3-1, TerraFly is shown with its basic navigation map through which users can request information from different addresses. Figure 3-2 shows the basic interface of TerraFly real estate application.

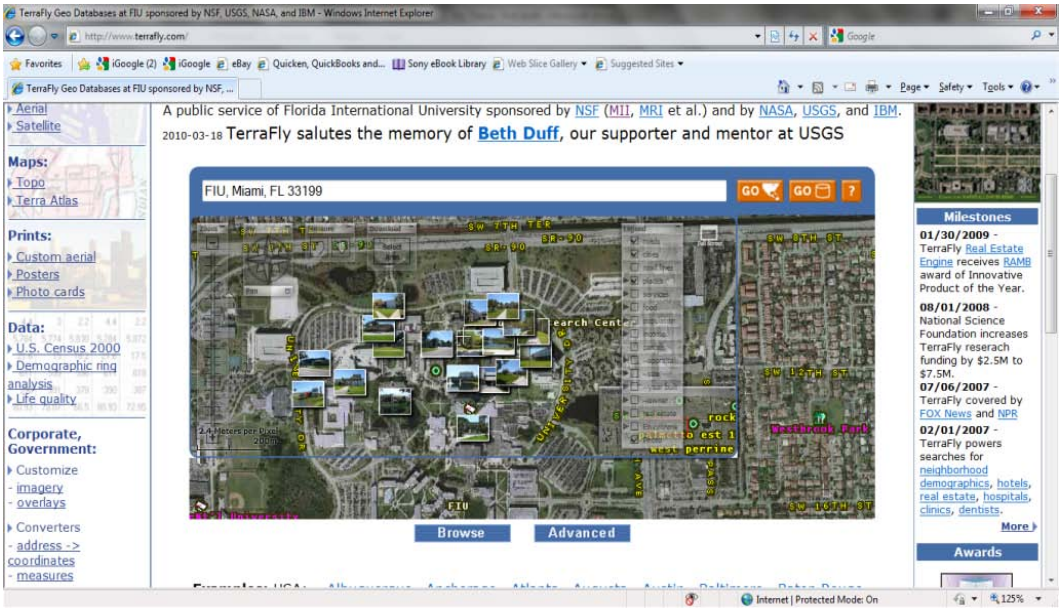

Figure 3-1: Basic TerraFly interface which shows a navigation map through which users can jump from one area or address to another. For each area, different information can be summarized and produced. Social information includes residents' average income and total population. Geographical information includes cities, roads, and services around that area, places and more. Source: http://www.terrafly.com

links to locations & details	Photo	Bed rooms	Area total (Sq.Ft)	Address	List Price (US$)	SqFt Living Area (SqFt)	View	Status of listing	$ / sqft / list (US$)	Price / Market County Value ratio
1: 163' $171K 1/1		1	950	9195 COLLINS AV # 910	$171K		Ocean	Active		0.829
2: 163' $100K 9195 COLLINS AV # 405				9195 COLLINS AV # 405	$100K	520	Ocean / Intracoastal	Active	$192	0.746
3: 163' $189K 1/2		1		9195 COLLINS AV # 1006	$189K	1050	Ocean	Active	$180	0.796
4: 353' $200K 2/2		2		9273 COLLINS AV # 205	$200K	1173	Ocean / Ocean-direct / Water	Active	$170	0.8
5: 363' $129K 1/1		1		9172 COLLINS AV # 303	$129K	763	Garden / Pool	Active	$169	0.741
6: 363' $165K 2/2		2		9172 COLLINS AV # 401	$165K	984	Garden / Pool	Active	$168	0.744
$175K		2		9172 COLLINS	$175K	997	Ocean / Other	Active	$176	0.785

Figure 3-2: TerraFly table showing application specific information and in this picture the real estate application. Detailed information about each property is retrieved including price, total area, features included, photos and more. Source: http://www.terrafly.com

The TerraFly group has created a Sem-SQL as a semantic interface for SQL databases which translates Semantic SQL queries to its equivalent relational SQL queries [FIU10]. The interface adds a semantic layer so application developers can express their queries in a shorter form by reducing the complexity of the relational database operators. Complexity can be reduced by using very little information about the database schema and not the details like primary keys and foreign keys. Figure 3-3 shows the relation between legacy applications, new applications and the semantic wrapper.

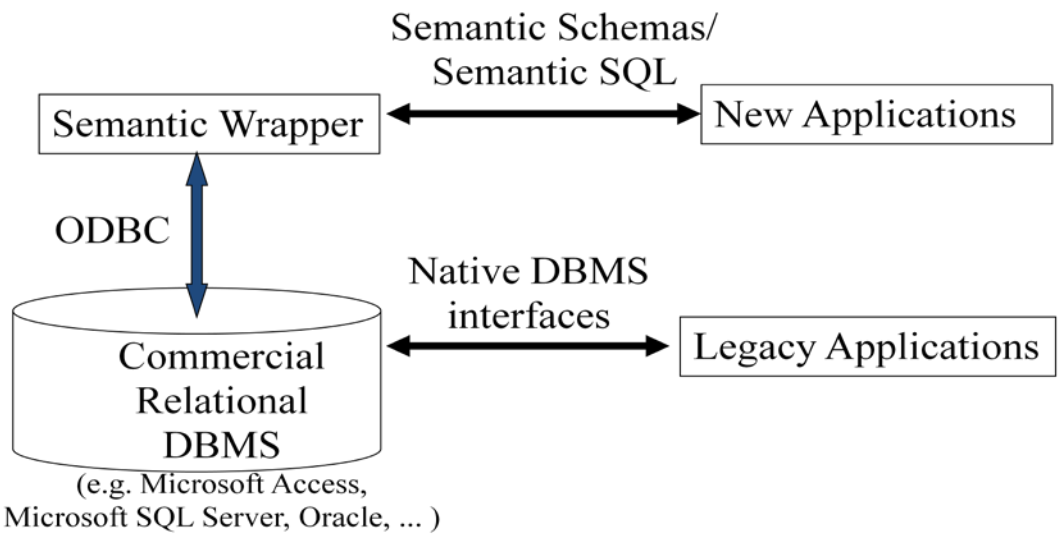

Figure 3-3: General architecture of TerraFly semantic SQL wrapper. Source: [FIU10]

3.2 Limitations:

- TerraFly uses a huge amount of data and deals with many partners. Each partner supplies data updates to TerraFly databases using structured but non-semantic data formats like XML. For example, real estate properties need to be updated regularly in case of a change in their statuses, prices, owners or any other information that is updated in the real estate offices and needs to be reflected on TerraFly databases. These data formats require programming efforts to update TerraFly and to make them available for use by other partners.
- TerraFly with its current model does not define ontologies through which it can extract meanings of its data and integrate it with other knowledge bases for a richer and more diversity layer of geographical information. Therefore, there is no interpretation of the words and phrases of the user query when the user searches for a place or an address.

Rather, it requires one or more components of the exact address which are not typically memorized by users.

- Reasoning on data without rules and semantics is not possible because semantic reasoners need description logic statements that define a set of rules.
- The current service requires a component of an official address like state, city, zip code, country or geo-coordinates. It lacks a mechanism to search for places without remembering address components.
- The current service makes searching for more than one geo-entity a hard job as it restricts the search for a specific location and then it ranks other retrieved results according to its own criteria without considering or involving the user. The semantic Web approach gives a better opportunity to the user to give his/her selection in advance where he/she wants the results to be ranked according to the closest place to his/her location or to the closest type to his/her entered entity type.
- TerraFly has developed a semantic wrapper which acts as an improvement for the current relational database query. Although the Semantic SQL offers several advantages like simpler application design and shorter programming cycle, it provides semantics between tables (not the data). So, a user asking for a dentist who is located within 10 miles of Manhattan will not benefit from this wrapper as it is not able to understand the details of his request.
- A schema re-engineering process is required for other knowledge bases to be able to use the semantic wrapper.

3.3 Search Service Enhancement:

The search service component in TerraFly aims to find places and locations of interests to users. It has a high potential opportunity of enhancement for the following reasons:

- Adding a semantic dimension to TerraFly requires RDF-based data that can be obtained from semantic data sources on the Web such as DBPedia and Geonames. Moreover, some knowledge bases already have developed SPARQL end-point interface through which client applications can pass a query and receive a result set formatted in RDF format. The available portals that connect several data sources are a foreshadowing of the future Web of linked data.
- Disambiguation usually arises from either an integration of heterogeneous knowledge bases or from misinterpretation of a human request .The former has another opportunity for a wider enhancement to TerraFly which is out of the scope of this thesis and the latter was selected to be the candidate as it relates to the research questions and business needs.
- Since extracting hidden knowledge that is relevant to the user query can best be evaluated by the user him- or herself, developing an interactive component with the user will result in a better evaluation.

Chapter 4. Motivation

The growth of the Web due to increases of network speed, number of internet users , storage capabilities and other internet technologies, has suggested there would exist a much higher potential for exploiting the Web if tools were available that better match human requests [Ege02b]. The geospatial domain is a challenging area due to the variety of data models, formats, semantics and relations. Moreover, reasoning with geospatial information would benefit users by providing a richer set of information that they do not have to explicitly ask for in their queries.

Current geospatial search engines do not provide users with the flexibility they need in order to express their queries with their familiar vocabulary. For example, users have to adhere to specific terms that are explicitly defined in the target databases. This is not practical especially for Web search engines, which are most typically available to public and general users. On the other hand, semantic search enables users to ask for queries where a composite query is divided into parts and each part is answered by a different data source. While combining the answers into one final answer might require expensive geometric operations because of the nature of the geographical information, semantic search provides a better approach. Semantic search provides dedicated tools and mechanisms to efficiently process rules that infer knowledge, and provide geometric comparison [ST07] that could contribute to the user query.

The nature of the graph pattern represented by RDF, OWL and SPARQL enables semantic geospatial search to accept users sub-queries defined in their vocabularies in a more flexible manner. This will increase user's acceptance to the semantic search. The required task will then be translating these vocabularies into a semantic query and process them where each sub-query

answers a separate question and probably from the same or different data sources. Combining the answers to meet the final solution of the user can be done with the help of ontologies as they facilitate the validation process.

The variety of geospatial relations like "*nearby*", "*close to*", "*in*", "*south of*", "*east of*" etc., was a motivator for defining an ontology in which the properties of these relationships better interpretat and accurate for geo queries. Also, it was necessary to approximate these terms with their possible contexts in order to deal with them correctly when they were encountered as properties of some resources in the knowledge base. Some typical examples considered as shown below:

Example 1: If the user asked for "*All shopping malls south of UM*", then we can instead retrieve shopping malls to which the *UM* entity has originated a relation *North Of*. However, such an assumption can only be made about the geographical relations if both *South Of* and *North Of* were defined as inverse relations to each other somewhere in the ontology.

Example 2: Consider the query, "*Entertainment centers west of Kendall*". While the "west of" relation has no exact boundaries, proposing a boundary by the system can yield a behavior that seems strange to some users. Therefore, a fuzzy [REW05] boundary can be used and applied where even those centers outside the boundary of *Kendall*, but very close to the west of *Kendall*, can be considered and given a rank value.

Another factor that motivated the design of a semantic-oriented geospatial search service was to improve the efficiency of processing the queries by limiting the search space. For example, a user might have asked for, "*All colleges in UM.*" Knowing that *UM* is a university can reduce the

search space drastically by retrieving all entities having a *College* relationship with the UM. This is in contrast to searching for colleges in a wider geographical area and filtering them based on which one is located in the UM.

Chapter 5. Related Research

5.1 Semantic Data Integration:

What makes the integration of geospatial data more complicated than other data is the variety of existing data models, formats and semantics, and spatial relationships. They are limiting factors to ensuring true geospatial interoperability [GDG10]. The use of semantic translators in dynamic approaches is a more powerful solution for interoperability than approaches that promote standards [Bishr 1997]. This section presents several papers found in the semantic Web research literature that address how computer scientists in the semantic Web community approached the creation, manipulation and integration of the geospatial ontologies and data from existing structured and unstructured data on several knowledge bases. Such ontologies were part of the efforts to smooth data integration and interoperability. In order to employ the Web as a medium for data and information integration, comprehensive datasets and vocabularies are required as they enable the disambiguation and alignment of other data and information [ALH09]. However, since most of the available semantic knowledge bases like DBpedia are results of a migration process from another data source, those semantic knowledge bases are not immediately ready to be integrated with other semantic knowledge bases. Novel methods and approaches were implemented to convey such integration processes. This is because the original migration processes were mostly implemented using information extractors like text parsers which resulted in inconsistencies and missing information especially that the geotagging process was not implemented perfectly as we will see later. The remaining of section 5.3 addresses some of these methods.

5.1.1 ODGIS (Ontology Driven Geographic Information System):

The aim of ODGIS [FEAC02] was to introduce an architecture that enables the integration of different system sources based on semantics and regardless of their representation. As ontologies are used dynamically by systems to generate implicit knowledge, they can also be browsed by system users to correlate their search terms with their related concepts and sub-concepts. This can be done either through generalization or specialization.

The proposed architecture goes through two stages: knowledge generation and knowledge use as shown in figure 5-1. Knowledge generation manually builds a formal model for the geographical concepts with their semantics into ontology and then translates this ontology into implementation classes like Java classes with their embedded functionalities and behaviors for system use at run time. The connections between the classes with the information residing in the knowledge sources are established through mediators [FEAC02].

Knowledge use is an interface that provides the user with a utility to browse the ontology classes in a hierarchical manner for the purpose of understanding the lookup concept with its connections, relations and specifications. Moreover, it provides the end user with a metadata information about the existing knowledge.

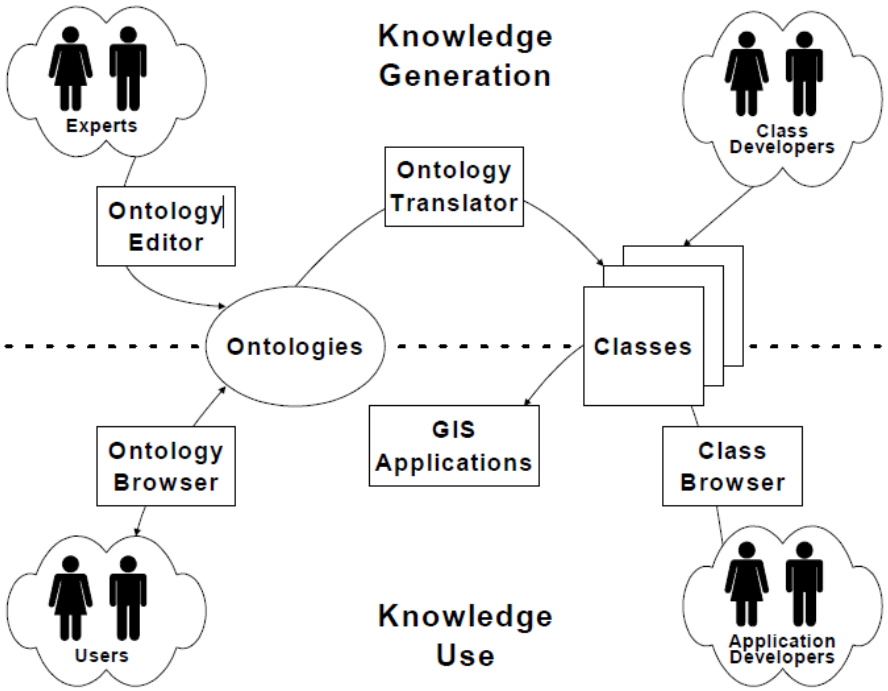

Figure 5-1: ODGIS general architecture. Source: [FEAC02]

5.1.2 **Ontology Based Markup of Geographically Referenced Information**:

Hiramatsu **[**HR04] has created two ontologies for the purpose of semantic utilization of the geospatial information on the Web. One describes the geo-feature classes and their relationships to other classes. The other one is to describe geospatial relations. Table 5-1 shows sample classes and properties of both ontologies.

Table 5-1: Geographic ontology types with basic classes and properties. Source: [HR04]

	Sample classes and properties
Geographic feature	Continent, Country, City (Shape) Point, Line, Polygon, MultiPolygon
Geographic relation	(Topological) Equals, Disjoint, Touches (Mereological) isWholeOf, isPartOf (Direction) isNorthOf, isLeftOf, isBehindOf (Space distance) withinMetersOf (Time distance) withinMinutesOf

Hiramatsu has also developed a plugin tool for JUMP which is a lightweight GIS for viewing, editing, analyzing, and processing spatial data [HR04], in order to create instances. These instances follow the feature ontology and apply geographical relations or connections between different spatial objects. They have also developed a relation calculator that discovers unestablished relations between instances based on their coordinates. However, since OWL is restrictive in expressing spatial relations, inference of qualitative relations is derived from coordinate based services.

Hiramatsu has followed a top-down approach which results in more data consistency. In our work, we integrate more classes and concepts to the basic ontology (DBPedia ontology) as we will see later, and we also build a subset of geo-relation ontology. However, since our relation ontology is not part of DBPedia ontology but we have created it to discover meanings of the geo-relations, these relations are not exist between the resources. They need to be calculated and established based on geo-coordinates. Alternatively, they can be retrieved from other knowledge bases. For example, both "South Miami" and "Miami Beach" are resources that exist in DBPedia but the relation that states "*Miami Beach is North of South Miami*" does not exist.

5.1.3 Toward the Semantic Geospatial Web:

Egenhofer [Ege02a] addressed the issue of whether the current semantic Web approaches are sufficient to answer user query and especially geographical requests. Many approaches are trying to represent the meaning of the query by using some primitive objects and relations in order to create more complex objects and relations. Moreover, many geographical data require that some primitive objects are combined together in a meaningful way in order to satisfy the user request.

Egenhofer addressed two main issues to solve and he defined a framework towards a reasonable solution:

- No standard form exists through which we can send a geographical request and interpret the request correctly on the other receiver side.
- No methods exist to evaluate the semantics of the available data sources in order to decide if their structure can be used for any particular geospatial query task.

Egenhofer proposed a solution for a meaningful geospatial info retrieval from the Web in the form of a geospatial request as formalized below:

<geospatial request> ::= <geospatial constraint>

[<logical connective> <geospatial request>]

where multiple geospatial constraints can be part of conjunctions or disjunctions using the logical connectives "*and*" or "*or*".

Each geospatial constraint is made up of three parts—two geospatial terms that are linked by a geospatial comparator as shown below:

<geospatial constraint> ::= <geospatial term>

<geospatial comparator>

<geospatial term>

Each geospatial term is either a feature class like "hospital" as defined in the underlying ontology or a label like "Miami" that is referring to a geospatial gazetteer. Moreover, the semantic of the comparison operator is described in the geospatial-relation ontology used. (e.g., *in (a, b)* ←→ *interior (a) isPartOf (boundary (b) Union interior (b)*).

The framework defined by Egenhofer is reasonable but it lacks the compatibility with the existing and already approved semantic Web standards like SPARQL. In order for us to be able to use this framework we first need to either replace the SPARQL query with this request format or to pass such a request form along with the SPARQL query. In our case, we are not able to either apply any of the two because we do not control the DBPedia knowledge base but we use it as a Web service. So, even if we passed the request to DBPedia, we need an interpreter to evaluate this request format and translate it into a language that is understood by the knowledge base in order to use it to search for the desired results.

5.1.4 Linking Open Data Project (LODP):

This project aims at applying the best practices in order to achieve more data integration on the Web of Data using Linked Data principles. The LODP vision is to discover the open license data sets available on the Web convert them into RDF and publish them [BHBL09]. Since the Web has changed from a world of connected documents to a Web of connected documents and data, Linked Data works on publishing and interlinking the structured data from different domains like people, companies, books, scientific publications, films, music, television, and radio programs, genes, proteins, drugs and clinical trials, online communities, statistical and scientific data, and reviews [BHBL09]. Linked data can be established between any different data structures on the Web like two databases in one or two different organizations, heterogeneous data sources and so on provided that this data carry its meaning, which is an essential part of its integration. The linked data principles were published by Berners-Lee (2006), and are used as guidelines for publishing data on the Web. They are briefly:

- Assign a URI for every object on the Web.

- Use HTTP URIs in order for people to be able to retrieve those objects through the Web.
- When an object is called, provide semantic information using the standards like SPARQL, RDF, OWL and ontology reasoners.
- Together with passing the called object, pass URIs so people get more useful and related data objects.

Moreover, the Web of Data was defined and given the following characteristics [BHBL09]:

- The data must not contain information on how to represent them.
- The data must be self-describing and it must provide links to its definition if an application was not able to translate the data vocabulary.
- Using HTTP and RDF provides a better mechanism for smoothly linking Web data compared to the application provided interfaces.
- The Web of Data is open, which means that applications do not search in a static limit of data sources but can be extended at run time by using links.

The nature of the LODP allows anyone to publish data on the Web adhering to the Linked Data principles and interlinking it with existing data sets. The progress of LODP as of July 2009 is illustrated in figure 5-2. Nodes represent the published datasets and the arcs between nodes represent the links between the different data sets. According to the LODP community, the Web of Data consists of 6.7 billion RDF triples that are connected and linked by 142 million links as of July 2009 [BHBL09].

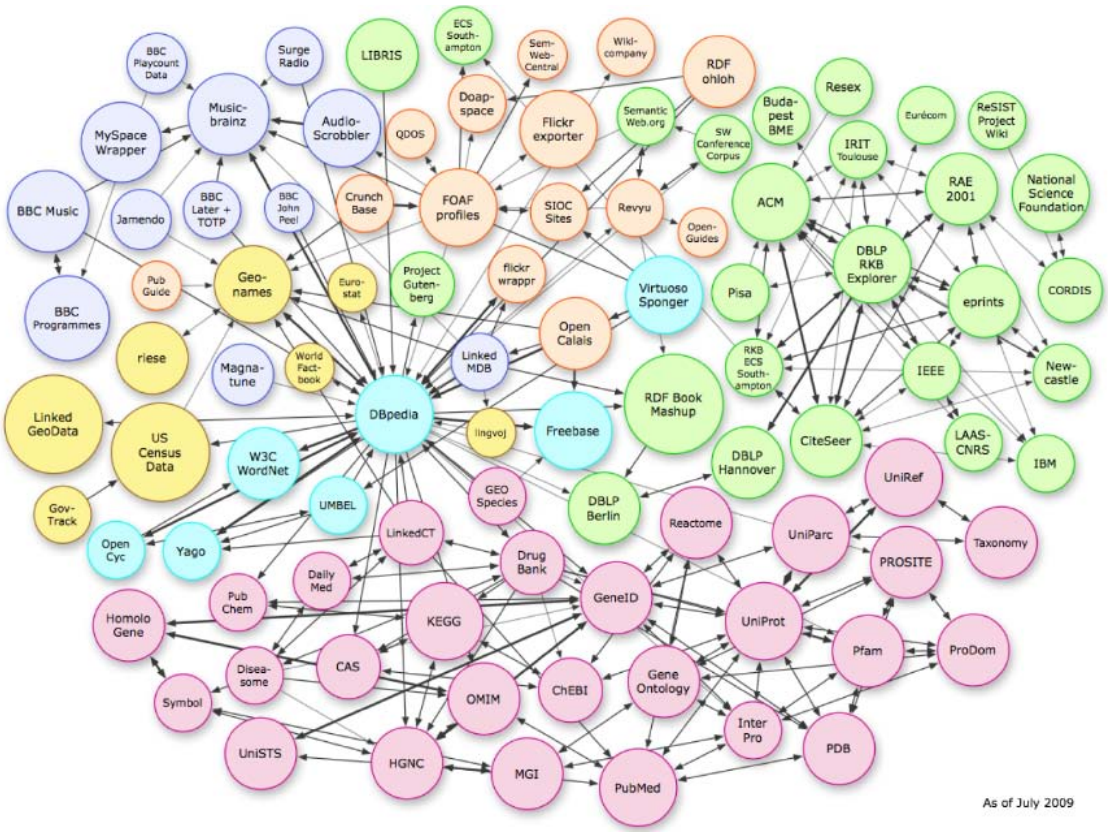

Figure 5-2: LODP as of July 2009 showing the integration of the different open data sets on the Web. Source: [BHBL09].

Publishing the data on the Web of Data as part of the OLDP requires the following steps:

- **Assigning URIs to the resources in the participating data source**: This is based on selecting unique HTTP URI for a given resource in a specific data source. Since there are many resources named differently and assigned different URIs by different data source owners, *OWL:SameAs* is used. It is used to link those resources which are the same but named differently by different datasets. This is very common in linking geographical information. For example, both DBPedia and Geonames defined and published the geographical resource "Miami", which is one of the largest cities in the state of Florida in the US. However, DBPedia has assigned "Miami" the URI

http://dbpedia.org/resource/Miami while Geonames assigned a different URI to the same resource *http://www.geonames.org/4164138/miami.html*. OLDP suggested that to overcome this problem each of DBPedia and Geonames can use the *OWL:SameAs* property in order to map both URIs.

- **Linking the resources with other useful resources on the Web using RDF links**: In many domains, there are identifiers through which linking data can be done easily. For example, in the library domain it is possible to link a list of author books from one library to another using the ISBN numbers. However, many other domains do not have identifiers and therefore heuristic approaches are required, supported by similarity computation algorithms. The Silk [VBGK09] framework was developed and used to compare different entities residing on different knowledge bases and therefore there is no need to replicate the data but rather to use the online SPARQL endpoint with Silk to link the different data sources.
- **Developing and publishing metadata sets that describe the published data**. This is important in order to provide clients with information that will help them use the data residing in a knowledge base, and how this data is interlinked by some relationships with other data on the Web.

To conclude, LODP is an effort supported by standards and a shared agreement between many different parties for the purpose of designing a global interlinked Web of Data. Although LODP has defined a procedure in order to publish and participate in a shared world of connected data, it has defined no mechanism on the details of how to search this data efficiently, which is critical to this research. Assigning URIs, linking them to others and describing the referenced data were all

addressed but applications still suffer from precision due to the lack of searching standards. SPARQL was a very general standard for all types of applications that support the search engines. However, SPARQL works perfectly when there are not any inconsistencies in the knowledge base. Inconsistencies greatly affect precision.

5.1.5 **Geospatial Resource Description Framework:**

Alam [AKT08] , has proposed new schema constructs that translate the GML (Geographic Modeling Language) constructs into OWL called, GRDF (Geographic Resource Description Framework). “Because of the world-wide adoption and standardization of GML, GRDF is designed to match GML in its content descriptions and feature relationships” [AKT08]. GRDF is based on a subset of first-order-logic, and it defines a language that can be used by different domains to create their own ontology for the purpose of better interoperability and more specialized and restrictive geo-language. The final aim is to develop extendable mid-level geo ontologies so domain experts can use them and extend them if necessary. Extending the ontologies will finally result in a low-level ontology that is domain specific, but understandable by other domains using GRDF. The GRDF ontology is shown below. While the feature model describes the different concepts in the geo-domain, the geometry describes the geometric shapes.

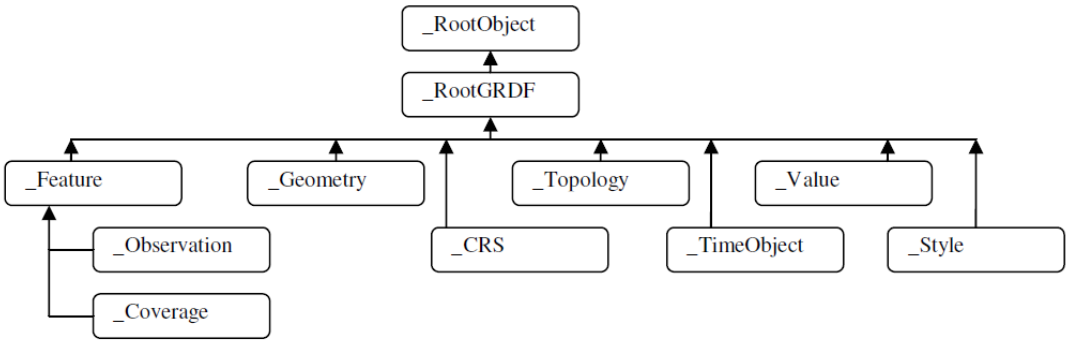

Figure: 5-3: GRDF ontology architecture. Source: [AKT08]

The creation of GRDF ontologies is an effort towards another and more specific layer of geospatial language standardization and towards the geospatial semantic Web. One important advantage of GRDF is its mapping from GML which might enhance and speed up the conversion of GML based data sources into semantic or OWL based data sources. However, its success depends on its comprehensiveness and popularity. While the first one can always be improved, its popularity and acceptance in the semantic Web community needs powerful evidence to support its need. While it is true that more geo properties and concepts have been integrated into the main basic ontology, GRDF need not be used but rather the basic ontology properties need to be used because the ranking was evaluated based on the instance representation of the underlying ontology.

5.1.6 Building Place Ontologies for the Semantic Web:

Abdelmoty [ASJ07] has identified some limitations of the ontology defined by OWL in representing the geo-place ontologies and he defined two alternative frameworks. The following OWL limitations were recorded:

1- OWL is not suitable for constraint checking since it describes first-order and open-world semantics.

2- OWL lacks the ability to express the inference patterns of a triple form, which is important in reasoning geospatial rules. For example

$$\forall x, y, c : \mathrm{rel1}(x, y) \wedge \mathrm{rel2}(y, c) \rightarrow \mathrm{rel3}(x, c)$$ [Hor05]

3- OWL uses generic data types for the geospatial data, which leads to a poor maintenance of that data.

4- OWL does not support computation in order to perform even basic geometric calculations like computing areas of a geospatial object.

5- There are high memory requirements for the geospatial location storage and reasoning which might be best done over a database system.

To overcome the above limitations two systems were proposed, the Centralized and Distributed approaches to ontology development.

- **Centralized Place Ontology Development**:

  It is based on maintaining the geospatial concepts, relations and instances in a central store, and adding more data originating from different knowledge bases. Two stores are used in this approach, the geospatial terminology and relations store, and the location and geometry data store. While the first store deals with the semantics and hierarchical structure and relationships in an OWL ontology, the second one handles the storage, indexing and manipulation of place geometric locations.

  Reasoning over data is possible using a spatial reasoning system which performs basic and complex geometric validations as if a polygon is closed and has at least three different points in space. In addition, relationship reasoning is possible here when, for example two cities are connected with a relationship "*locatedIn*" implies that one city is larger than the other. However, this kind of reasoning produces qualitative measures only.

- **Distributed Place Ontology Development**:

  Rather than storing place information from different data sources, the distributed approach is based on accessing information from different external data sources as required. Most geographical data sources produce metadata describing the characteristics of their data source, which can greatly help in deciding which data source should be accessed online. A search request is initiated by the place ontology towards the available data sources. The results are then merged and ranked.

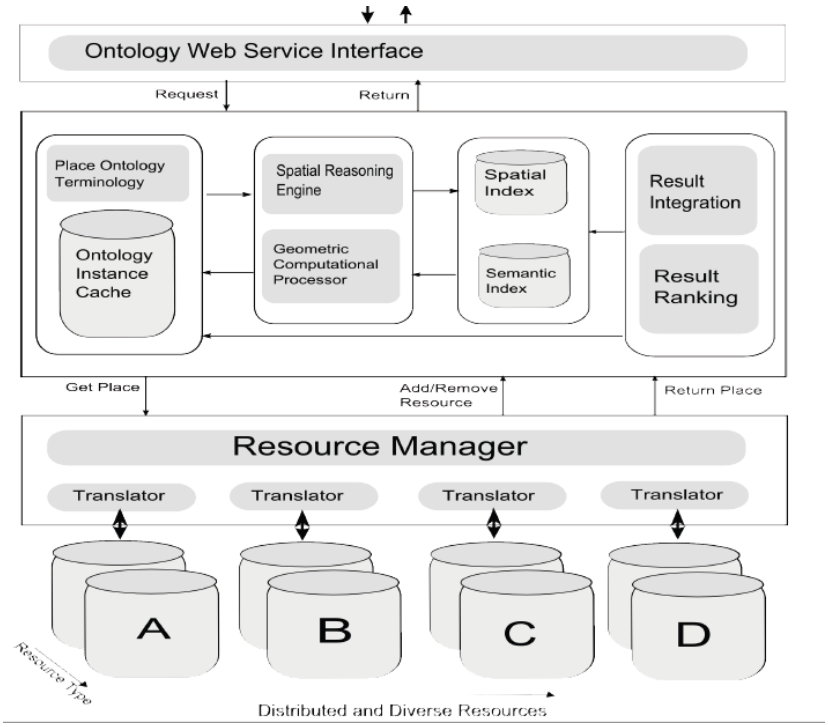

Figure 5-4: Distributed place ontology approach. Source: [ASJ07]

Part of both approaches was used. The first approach was followed in terms of maintaining the geospatial ontology, terminologies, and semantics in the local system and using them to retrieve information from other data sources, which gives narrower and more precise results by following

part of the ontology structure. Second, external data sources like DBPedia and Geonames were used to retrieve the place information and then rank them based on ranking scores.

5.1.7 A Semantic Search Engine for Spatial Web Portals:

The goal of *Wenwen* was a semantic search engine that integrated information from different Spatial Web Portals like NASA's Earth Science Gateway (ESG), ESIP's Earth Information Exchange (EIE, http://eie.cos.gmu.edu), FGDC's Geospatial One Stop (GOS, http://gos2.geodata.gov) Portal, NASA's Earth Observing System Clearinghouse (ECHO, http://www.echo.nasa.gov/), NASA's Global Change Master Directory (GCMD, http://gcmd.nasa.gov/), and NOAA's National Climatic Data Center (NCDC, http://www.ncdc.noaa.gov) [LY08]. A major problem with the traditional structure of using an ontology in semantic search engines is the separation of the ontology TBox from the instances ABox, which makes the reasoning process possible only on the first one but not the second. Initially, the SWEET (Semantic Web for Earth and Environmental Terminology) ontology which represents a TBox of the geographical resources was integrated with the ABox instances to overcome the reasoning problem.

Wenwen is proposing a sequence of operations in order to achieve semantic reasoning to process the search query. The sequence is syntax analysis, semantic analysis and retrieval tasks. Syntax analysis is parsing a user's query in order to identify the critical terms. Some natural language parsers were developed and used. Alternatively, the search query can be more controlled by providing a template to the user, which he/she can follow thereby obtaining no disambiguation about the query place or location terms. Semantic analysis concerns reasoning over the TBox and

ABox in order to get the semantics of the parsed query terms, and then using these semantics with the original user query in order to construct the final query. Finally, the system sends the constructed query into different portals using what is called the "Geo-bridge" [LY08], which either sends the query request using the portal API if exists or uses a portal developed interface like XML based Web interface. Extractors were developed in order to extract information from the returned results, which can be a Web page.

The same general architecture was used in the semantic search service. However, the target data sources were semantic data sources from which it was possible to accurately and precisely extract information from the returned results. Moreover, the model does not need information extractors from the returned query because it is know which objects to retrieve prior to sending the query. This is an important characteristic of the semantic Web over Web 2.0. A user query template was used that restricted user entry but resulted in better and exact interpretation and separation of the user place, relation and location terms.

5.1.8 Open Street Map:

The Open Street Map (OSM) was established with the aim of encouraging the growth, development and distribution of free geospatial data and providing geospatial data for unrestricted public use and sharing [Wik10]. While the Open Street Map project is currently to render maps on the Web, it has collected more data than many commercial geo-products due to there being more contributors in some regions than in others. The OSM has achieved some integration with the Geonames knowledge base by matching a user request with the closest geo-entity from OSM and Geonames. However, we found that the OSM is not always able to specify

the exact location of the returned Geonames result but it was possible to achieve that using the Geonames location identifier and the hierarchy Web service from Geonames.

Additionally, it was found that many of the OSM data lack semantic or misspelling anticipation , e.g. if the user entered "Dadeland Mall" searching for one of the shopping malls in Miami, the OSM returns no results because this mall was annotated with a missing letter at the end to be "Dadeland Mal" but the OSM search tool ignored the semantic orthography of the user query. This research included a spell checker library which added a semantic dimension to the user query before processing the SPARQL query.

Since the OSM has accumulated a huge amount of geospatial data and integrated it with some knowledge bases like Geonames, it is a rich data source. However, it lacks some semantics capabilities since OSM data are XML based. The linked geo-data (LGD) project was aimed at solving this problem by mapping OSM to RDF triples. The LGD project is discussed in details in the following section.

5.1.9 Linked Geo Data:

The Linked geo-data project aimed to transform the OSM geospatial data into a pool of semantic and more integrable data. This was done by mapping the OSM data from XML into RDF triples, and establishing links to other entities on the data Web. Currently, LGD has collected more than 3 billion triples [ALH09].

LGD has borrowed some of its ontology structure from the OSM relational database as in using the class *geo-wgs84:SpatialThing* with subclasses node, way, relation and properties such as *geo-*

wgs84:lat and *geo-wgs84:lon*. Moreover, users can create their own attributes, name them and assign values to them. This is the reason the LGD community standardized these attributes by categorizing them into 3 main categories, which also helped in building the ontology that contains 500 classes. These categories are namely the classification, description and data attributes. The classification attributes explicitly specifies if the current element is a member of another class. Using these attributes, a hierarchical structure could be built that links classified objects with their class with an "*Instance of*" relationship in the ontology. The description attributes describes other values of the element like latitude and longitude, and finally the data attributes which are storing the user free text descriptions [ALH09].

Part of the LGD project was publishing data using the Triplify [ADL+09] approach that publishes data from relational databases by mapping URL requests into relational queries. Moreover, Triplify can search for locations taking the current latitude, current longitude, radius to search for and the filter that return the results with a specific property or property value [ALH09].

For geospatial data interoperability and integration, the LGD has implemented a mechanism to map to the DBPedia resources which consisted of schema mapping and entity mapping. The schema mapping between LGD and DBPedia was partially met using the DL-learner tool [LH08], and the entity mapping was met by deriving a matching heuristic based on three criteria. First, the type of information (e.g. checking whether two objects both correspond to cities) is derived from a static table that defines a mapping between the DBPedia and LGD types. Second, the spatial distance is based on ranking the LGD entities that have the closest distance to a given DBPedia entity. Third, name similarity was used to compute the similarity of scores of the

closest LGD entities using a Jaro distance metric [BB09]. However, Many DBPedia entities, which cannot be matched, do either not exist in LGD, are not classed, or are misclassified in DBPedia [ALH09].

The LGD project has also implemented browser and editor interfaces through which users can browse and update the LGD. When changes are made to the geographical entities using the LGD editor, these are stored locally and propagated to the main OSM database by using the OSM API [ALH09].

The investigation about LGD can be summarized by these points:

- Using the classification attributes to create the ontology can help in building the class relationships. However, in order to build a comprehensive description of an entity and from the definition of the ontology, other relationships are needed between classes which some-how relate one class with another. Moreover, restrictions such as cardinality and range values are also part of the ontology which can improve the description of the domain entities. In this research, a *Directory Ontology* was built which described such a hierarchical structure but integrated other ontologies like DBPedia as an effort to be as comprehensive as possible.
- The LGD has implemented a semantic matching mechanism between the LGD and DBPedia schemas using a DL learner. Since in this research geo-data was not maintained but rather different information from different data resources at run time was integrated. It was more appropriate to use the ontologies defined by the data sources. Moreover, the Jaro distance, semantic terms and spatial distances were used to match the different geospatial resources with the user query semantics.

- It has been reported for LGD that there were some unmatched entities between LGD and DBPedia due to some entities either not having been classified or having been misclassified. However, since the research is focused on semantic geospatial search service, the approach to DBPedia data was different through evaluating as many attributes as possible from the single entity in order to match it or classify it. Such an entity would only be processed if it was identified as a geo-entity. For LGD only those entities which describe the latitude and longitude values from DBPedia were considered in order to process the mapping between the LGD objects and DBPedia geo-entities. However, there are many other attributes in DBPedia entities which can also be used to identify the geo-entities like *dbo:location* and http://www.georss.org/georss/point. These attributes among others were used in order to retrieve the relevant geographical resources and this sometimes did not help in determining the exact location of the resource but it could at least retrieve it if it existed.

5.1.10 DBPedia Mobile:

DBPedia Mobile is a cell phone semantic Web application that renders a map indicating nearby locations from the DBPedia dataset. It also enables users to publish their current location, pictures and reviews to DBPedia and interlink them with a nearby DBPedia resources thereby adding a rich layer of geospatial semantic Web data. The DBPedia was selected since there are approximately 2,180,000 data links pointing from DBPedia into other RDF data sources on the Semantic Web [BB09]. Therefore, DBPedia mobile used the RDF links to other datasets on the Web to provide the user with richer navigation by including Web links that contains more

information. *Owl:sameAs* and *rdfs:seeAlso* are followed for up to two levels in order to gain more information about the resource, and to obtain human-friendly resource labels [BB09].

DBPedia Mobile is a client-server application. The client side is written in JavaScript and the client location is estimated using a combination of the IP address and Yahoo! FireEagle Web service. The server is implemented as a Java Servlet using the Sesame RDF framework. MySQL Spatial extensions were used to support the storage and filtering of geospatial points [BB09].

Since the entity interlinking using the *SameAs* and *SeeAlso* only provides the same geospatial result but with more information, the semantic search service priority is to provide richness in terms of the resource varieties rather than the resource richness of information.

5.1.11 DBPedia Navigator:

Because users are not able to express their queries in a structured form, DBPedia navigator offered a navigation suggestion mechanism using machine learning techniques. Furthermore, it provides many other means to browse through the DBPedia knowledge base taking the underlying semantics into account [LK08].

DBPedia Navigator uses a supervised symbolic machine learning method to provide the user with navigation suggestions relying on the semantics of the underlying data. It uses inductive learning to find logical explanations for given data. Several class descriptions are tested and evaluated during the learning process. Since the evaluation process is done using an OWL

reasoner, which is impossible to handle for such a big DBPedia knowledge base, only some fragments are extracted from DBPedia. The extraction works by executing SPARQL queries, which obtain knowledge related to the example instances [HLA09]. After evaluation, smart algorithms will further use these evaluations to suggest more descriptions. So, the learning process can be described here as a search for the best description. The well-known strategy called ILP (Inductive Logic Programming) was used to build a top-down refinement operator based algorithm. This algorithm is used to test the most general description, then mapping that to more specialized classes that cover all the positive examples [LK08].

Our research extracted semantics rather than build a knowledge base which would have been too costly in time and effort.

5.1.12 YAGO-NAGA:

YAGO (Yet Another Great Ontology) is automatically building a large, highly accurate and searchable knowledge base by applying information extraction (IE) methods to Wikipedia and other sources of latent knowledge. The main goal of YAGO is to build a comprehensive knowledge base that knows all individual entities of the world, their semantic classes, relationships between entities and to be capable of logical reasoning and query support. YAGO primarily gathers its knowledge by

integrating information from Wikipedia and WordNet and currently contains 249,015 classes, 1,941,578 individual entities, and about 19 million facts [KRSW08].

Information Extraction:

YAGO does not apply pattern, NLP and learning-based IE techniques on texts to extract information due to a higher risk of low precision. Instead, YAGO uses a text-oriented harvesting by applying two information extractors:

- The infoboxes: YAGO uses Wikipedia infoboxes in order to extract their attributes and map them into an RDF triple. It uses rules for frequently used infobox attributes to extract and normalize the corresponding values. [KRSW08].
- Wikipedia Categories: YAGO uses the categories to build the "*InstanceOf*" relationship between the entity and the category that it belongs to. Some entities do not apply the "*InstanceOf*" relationship with their categories. Therefore, YAGO uses heuristics like linguistic processing to discover such relations.

YAGO Temporal facts:

YAGO added its own temporal facts to the extracted facts from other knowledge bases like Wikipedia. This was useful to indicate the range of date and times in which these facts were valid [KRSW08].

Using Wikipedia categories to build the relationships between classes and instances resulted in inconsistencies. Discovering all inconsistencies is not an easy job especially with a knowledge base of billions of attributes and properties like Wikipedia. Moreover, using these categories to build the geo-ontologies in question would be too complex and time consuming. This was true especially that recently a query found the number of geo-categories in DBPedia to include more than 1,455,844 categories.

5.2 Semantic Ranking:

The ranking of the results is a key problem in every Web search service, especially those required to crawl a huge amount of data deployed in many knowledge bases. In 2005, Google claimed to have indexed 8,058,044,651 pages. It is common that even the most famous search engines return results that are useless to the user. This is mainly due to the fact that "the very basic relevance criterions underlying their information retrieval strategies rely on the presence of query keywords within the returned pages" [LSD09]. Moreover, the problem becomes more complicated when the search engine is required to rank not only in accordance with the syntax of the user query and how the search results match the exact words, but also according to the semantics of the user query words, relations, and intentions. While the latter approach produces more relevant results because it interprets the semantics of the user query, ranking becomes more complicated since the ranking space becomes bigger.

The following are some related approaches that were applied to different semantic and non-semantic search engines. Most of the non-semantic ranking approaches were adopted by some semantic search engines. For example, PageRank [PBMW99] algorithm was applied by the semantic context-sensitive ranking in [Hav03b].

5.2.1 Ranking using SVM classifier:

Albert Bifet and Carlos Castill used a binary classifier [BCCW05] to group results into the top and the bottom of the ranking list. The classification uses the page features. Therefore, "given a

pair of documents, we try to predict which one is ranked above the other". This gives a partial ranking. Then, they used a SVM linear classifier to obtain the normal vector, and they used the scalar product to obtain the complete ranking. Another linear classifier was tested based on logistic regression using the probability of an instance to belong to a class given the instance vector value. In logistic regression, the glmfit (generalized linear model) with a binomial distribution from Matlab was used to compute the logistic regression models. If we have two classes A and B, given a coordinate x, the classification function is defined as:

$$\log\left(\frac{P(\,class=A|X=x)}{P(\,class=B|X=x)}\right) = \rho + w.x \quad \text{[BCCW05]}$$

Where ρ is an offset value and w is the weight for the factors used as a ranking criteria.
In SVM, different training data sets were used to train and learn the linear scoring function or a decision tree. The binary classifier works using the assumption that we have a vector of features where each page represents a vector value. If v and u are vectors and we are trying to find a classification function f such that $f(u) < f(v)\ if\ u < v$ which means that we train our model to find f based on our desired ranking which states that v is given a better rank than u. We then use this function to classify the real time instances based on the trained model.

$So, if\ function\ f\ is\ linear\ then$

$f(u) = wu$

$So, if\ f(u) < f(v)$

$$=> wu < wv$$

$$=> w(v-u) > 0$$

Clearly, we can see that the binary classification problem is the problem of finding a hyperplane with normal *W*, which perfectly separates the positives from the negatives. This is accomplished by building a classification tree through a process known as binary recursive partitioning. This algorithm iteratively breaks up the results into two groups by considering the value of one feature in each iteration.

In the geospatial semantic search there are many crucial features are hidden and cannot be observed especially when we deal with inconsistencies in the search knowledge base where a geographical entity has a set of features and the same type of that entity has only a subset of the features. This kind of inconsistencies could result in a high misclassification curve.

5.2.2 PageRank Algorithm:

This is a link based algorithm that is used by the Google internet search engine [PBMW99]. The algorithm is based on assigning an importance weight to a given document on the Web that measures its relative popularity compared to a set of documents. The numerical weight that it assigns to any given element *E* is also called the PageRank of *E* and is denoted by *PR(E).* Such a weight is calculated by adding the total subweights given by all the referencing documents to the document in question. Let us assume that we have a set of documents *A, B, C* and *D* and each will be assigned a PageRank score. Then, the PageRank conferred by an outbound link from a given document is equal to the document's own PageRank [PBMW99] score divided by the normalized number of outbound links *L*.

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}$$

Although the PageRank algorithm has proved its efficiency, some researchers claimed that many important pages that are not participating in the links party are actually ignored in this algorithm. Moreover, up to our knowledge, this algorithm does not consider the semantics of the user query but only the best syntactic match with the best PageRank score. Due to the nature of our search service which cares about the width (variety) of the results more than the depth of the result, we do not care about the popularity score. For example, a user is searching for "*Towers in Manhattan*" would prefer to see the list of all the towers without redundancy rather than the list of the most popular pages visited and which might contain information about some of them and ignore others. Finally, the PageRank algorithm might not be possible to be used in DBPedia because we do not deal with pages owned by different owners but with resources that can reference many important and non important resources based on the information extractors used to build the knowledge base.

5.2.3 A Context-Sensitive Ranking Algorithm:

This algorithm is based on the PageRank score. It computes a set of PageRank scores for each page offline, where each score represents the page importance with respect to a specific topic [Hav03b]. Using the Context-Sensitive approach, they avoided the problem of heavily linked pages getting highly ranked for queries for which they have no particular authority. Pages considered important in some subject domains may not be considered important in others, regardless of what keywords may appear either in the page or in anchor text referring to the page. By selecting the appropriate score that measures the page importance with respect to the

expected topic revealed from the user query, the algorithm is able to determine more accurately which pages are truly the most important with respect to a particular query or query-context.

The topic-sensitive algorithm works better when the number of topics is maintained to be relatively small. Two approaches are possible in selecting the different topics:

1- Clustering the pages into groups using machine learning methods.
2- Using a manual construction as a source of topics which in this project resulted in 16 different categories.

At run time, the context of the query is used to determine the topic by considering some of the words in the pages from which the user copied some of the query terms. If the query was not copied from any other source, a probability distribution is used to calculate the probability for each of the 16 topics based on the query terms. This could be achieved using the multinomial naive-Bayes classifier to compute the probabilities. Then, the URLs containing all the terms of the query are retrieved and ranked. The ranking is based on a ranking score that is computed as follows. Given a retrieved document d by processing a query Q with a category $Category_j$, score for *d* is calculated as:

$$score_d = \sum_j P(Category_j | Q) \times PageRank(d) \quad \text{[Hav03b]}$$

Our research approach calculates the rank of a result document based on a score as used here that takes into account several factors. However, where this approach considers PageRank with query

term matching, we consider other factors like the query term semantics and other factors that are particularly suitable to the geospatial search as we will see later in this research.

Although this algorithm narrows the ranking space by considering the topic sensitive pages, there is no reasoning and inference mechanisms through which we can evaluate how close this page is to the user query semantics rather than how popular this page is. The only other score used in conjunction with the PageRank is a similar approach followed by the traditional search engines using the keywords with little modification applied on the URLs.

5.2.4 Ranking Algorithm Based on User Attention Time:

This algorithm depends on the user profile history to make a user specific ranking. The algorithm compares the user reading duration for each document and if the duration passed a threshold value, then the system assumes that document is of more interest to the user [XZJL08]. For the next user search, the system will give a high rank to those documents that are of more similarity to the documents having more attention times from the previous search.

Although this approach tries to focus on the user by interpreting the user intention, it considers only one factor (reading time) and ignores many semantic factors that are related to the semantics and meanings of the user query. Moreover, this approach is more appropriate to the search engines specialized for retrieving Web pages rather than Web info because of its nature and dependency on calculating the reading time of the full opened page.

5.2.5 Ranking Complex Relationship Search Results on the Semantic Web:

This approach is different than the previous ones because it focuses on the semantic relationships rather than on the resource descriptions [AMS05]. It considers the problem of the ranking techniques used until now which return the same ordering for different query intensions without consideration of the user query purpose. It is also based on the user to specify a pair of resources and some keywords, and also to select the approach he wants with the degree of predictability. For example, if the purpose of the search is an investigation then the user can select the unpredictable relationships are to be ranked first. On the other hand, if the search was an ordinary one then the predictable relationships are to be ranked first.

This algorithm uses an S-Match value that specifies the degree of the relation match to the user entered keywords. The degree of a property matches and hence its S-Match value is determined by the closeness of that property and the properties in the property hierarchy. The hierarchical distance between a keyword k_i and a property p_i measures how close or far those objects are thereby contributing to the overall rank of the resources. The S-match is given by:

$$SemMatch(k_i, p_i) = 0 < (2^d)^{-1} \leq 1 \text{ [AMS05]}$$

Where d is the minimum distance between the properties in a property hierarchy.

SemRank is using the hierarchical distance that contributes to the final rank value of the result document. This has been applied to properties in links with concepts. In our research approach, we use the hierarchical distance as a preprocessing step to further add a semantic meaning and enrich the query terms with more meanings. This is achieved by considering the subclass/superclass of the place term entered by the user and also found in our geo-ontology.

Although testing this algorithm has shown its flexible ranking approach that provides a variety of result orderings that a user may choose from, it places more restrictions on the user query than we do. While our search is flexible in providing the choice to freely enter the geo-query, the SemRank applies a restriction on the type of search whether predictable or conventional which of no usage for the geo-search service. Moreover, SemRank uses are more applicable for applications like investigation, and we do not see its applicability to the nature of the geospatial search, which focuses on the resources more than on their relationships. However, we will follow the hierarchy distance approach to restrict our searching space.

5.2.6 A Relation-Based Page Rank Algorithm for Semantic Web Search Engines:

This algorithm follows a probability model to assign probabilities to the retrieved resources [LSD09]. Each probability measures how good a resource to the user by predicting the desired relation from the user query that exists in the ranked resource. This approach assumes that an underlying ontology is already exists, and the user will be given the freedom to select from that ontology whatever concepts related to his/her search. The search engine job will be in discovering the target relation between the user selected concepts and relying on the relation existence to rank the resources. The difference between the traditional search engine and this semantic engine is that the former will return the Web pages containing both words without taking care of the nature of the semantic relation between the two.

The domain ontology is represented as a graph where the nodes represent the concepts and the edges represent the relations between the concepts. The user query is represented as a subgraph of the ontology graph where the subgraph only contains the concepts specified by the user and are linked by a weighted edge. The edge weight measures the actual number of relations between two concepts in the underlaying ontology. Finally, each resource is mapped to the query subgraph and ranked accourding to its similarity to that subgraph.

Since we rely on semantic relations as part of our evaluation and ranking, we will partially follow this approach.

1- First, we consider how similar the result document to the concept specified by the user by examining how they match in terms of properties and relations. This is also known as the internal structure. However, the internal structure is only used as one of our ranking factors and is given a limited weight.
2- Second, we rank our resources based on their meaning similarity to the user query terms.

The differences between our approach and this approach are the following:

1- Our ranking is not based on the existence of predicted relations in the resource but is based on matching the concept type specified by the user with other factors.
2- We do not rely only on an ontology to create our query graph. We rely on WordNet to enrich the words' semantics and verify that they are geographical entities by examining the upper hierarchical classes.
3- We follow this approach because we provide the user with a less restrictive service so he/she can enter a free text query rather than selecting from the ontology concepts.

5.2.7 Hybrid Search and Ranking:

The hybrid search approach in [BCC+08] introduces a search methodology that combines a keyword-based approach with a metadata based approach. Several metadata based search engines have created an interface where users can browse ontology and navigate through the connected hierarchical concepts for the purpose of providing the user with description of the information stored in the knowledge base. The main reason for the hybrid search is because many of the underlying ontologies represent only part of the knowledge in the knowledge base, and therefore a pure semantic search is not enough. Another reason for the hybrid search is related to the keyword search which might misses to consider the meanings of the user query and therefore produces low recall. Moreover, ambiguity arising from the user query can result in misinterpretation of the user query terms and therefore produces low precision. Hybrid search overcomes the problem of low recall using keyword search and the problem of low precision by using the semantic search. In ranking the results, the hybrid approach follows the TF/IDF (Term Frequency/Inverse Document Frequency) only because the semantic ranking will assign the same scores for all the retrieved documents. This is because only those documents that match the ontology are retrieved from the knowledge base.

In our research work, we follow both the keyword and the semantic-based approaches. Semantic-based search is implemented to retrieve results that are close to the user query meanings and the keyword search approach is to overcome the inconsistency problem in the knowledge base. The inconsistency problem arises from the fact that some very important candidates are ignored during the search because they do not follow the underlying ontology concept. However, our

ranking is different from the ranking followed in the hybrid search. We have conducted experiments which proved that retrieving only those documents that meet the ontology reduces recall significantly in fact it sometimes ignores the best candidates. We therefore use a combination of ranking scores each represents a different aspect of ranking preference and some of these scores are purely related to geospatial search, like the spatial distance.

5.2.8 Discovering User Geo Intention on Web Search:

Xing [YRL09] focuses on interpreting the user's geo-query in order to improve the accuracy and ranking of the search results. A technique is used with the IP address of the user workstation to discover and disambiguate the location of the user, e.g. if the user query contains "*Miami*" does he mean the one in Florida, Ohio or Oklahoma? In addition, the term "localization capability" is used to define an approximate radius value and reduce or expand the search area around the user location so if the user searches for *"restaurant"* we might infer that the user will not travel as far as if he is searching for *"Car agent"*. The user query is divided into two parts namely "location part" and "non-location part" in order to further focus and simplify the discovery process of the user intended geographical location. The user intension is categorized into one of 3 categories namely:

- Local geo-queries: Are those queries with the user intension to find places or services within the local area of the user location. An example can be searching for a dentist.
- Neighbor region geo-queries: Are those queries with the user intension to find places in the neighbor region of his/her location. An example can be searching for a car dealer.

- Not easily localized queries which do not fall into the above two categories like searching for a hotel.

Xing [YRL09] also built geo-language models like the CLM (City Language Model) to discover users' specific implicit geo intents. Previous studies have shown that 82.77 % of the geo-queries routed on the Web contain information about the cities [JZR+08]. The CLM for example can help identify if the user query is at the city level by extracting those terms like 'map', 'hotel' or 'hospital' that are very frequently appearing in the city level queries. Such models were built using a large industrial-scale log for the YAHOO search engine. A statistical learning technique in [TK02] was used to learn part of this log in order to construct the CLM.

Xing's approach is meant to limit the search area, which we also apply in our system, but rather than deriving the search radius based on statistical learning, we further define the geographical relations. Relations are defined in our ontology in order to clearly employ their meanings when they appear in the user query. Interpreting the relation meaning has a very high probability of retrieving results that are undesirable but therefore we apply a further ranking mechanism which considers the spatial distance of the result entity. Moreover, we explore the non-location semantics of the user query in order to interpret the meaning of the user intension based on ontological processing and reasoning.

Although we apply a similar method used by Xing in splitting the user query into location and non-location parts for the purpose of identifying the user geo intention, we further restrict the

user query to a specific structure. This is possible in our situation because we only deal with geographical queries in fact the purpose of our search is to find a geographical entity.

Chapter 6. SGSS Design

In this section the concept and the design approach followed in order to achieve a semantic class geospatial search service. The design is in five stages: 1) integrating gazetteers, 2) geospatial ontology management, 3) geospatial resources discovery, 4) reasoning and 5) semantic ranking.

6.1 Integrating Gazetteers:

The location entered by the user in the query is the basis point from which was calculated the bounding box using the latitude and longitude values of that location. In order to obtain these values, a gazetteer was needed which was integrated from two sources namely the USGS (http://geonames.usgs.gov/) and ADL (http://it-drupal.library.ucsb.edu/adl-gazetteer). Both gazetteers were downloaded as text files, processed and stored into a database for fast location lookup during run time. The most important extracted data included feature name, feature class, state, county name, map name, latitude, longitude and elevation. At run time, an SQL query is passed to the MS access database in which the integrated gazetteer was saved and retrieved the best location match. A java class "Gazetteer.java" was developed to connect to the database and retrieved the required location data, which was faster than calling an online Web service. This class interacted with other geo-classes that were created for the purpose of interpreting the user query in a semantic manner and building a, *"Geo Query Object"*.

6.2 Geospatial Ontology Management:

6.2.1 DBpedia Ontology Examination:

The DBPedia ontology (T-Box) was tested and found that it provided a hierarchical and well-structured vocabulary to all the concepts in DBPedia and linked them to their object and datatype properties. However, because DBPedia instances (A-Box) were created by extracting the tags from Wikipedia infoboxes, and because those infoboxes did not follow a standard structure or ontology, but rather they were in XML-based format, there were many inconsistencies. For example, *Hospital* is a subclass of *Building* and the first one inherited all the properties as shown in table 6-1 from the second one and added more specific properties to them like "bedCount". It might be expect that any hospital instance in DBpedia would follow the same structure and would have values to all these properties, but in fact some are not even linked to these properties.

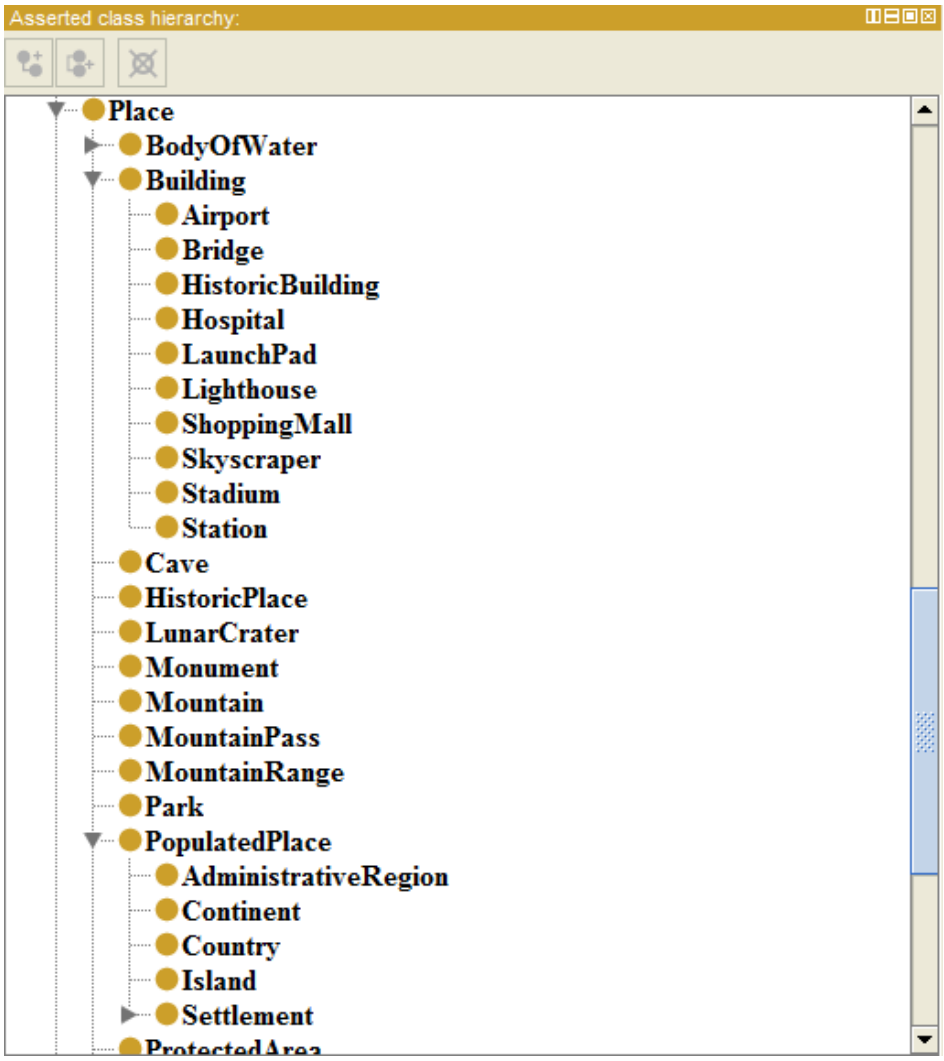

Figure 6-1: DBPedia upper level ontology displayed in Protégé.

Table 6-1: Properties of *Building* Class.

(http://dbpedia.org/ontology/structuralSystem)
(http://dbpedia.org/ontology/geologicPeriod)
(http://dbpedia.org/ontology/percentageOfAreaWaterRound)
(http://dbpedia.org/ontology/floorCount)
(http://dbpedia.org/ontology/depth)
(http://dbpedia.org/ontology/buildingEndDate)
(http://dbpedia.org/ontology/subdivisions)

```
(http://dbpedia.org/ontology/address)
(http://dbpedia.org/ontology/maximumDepth)
(http://dbpedia.org/ontology/visitorsPerYear)
(http://dbpedia.org/ontology/openingDate)
(http://dbpedia.org/ontology/demolitionDate)
(http://dbpedia.org/ontology/lowest)
(http://dbpedia.org/ontology/nutsRegion)
(http://dbpedia.org/ontology/plant)
(http://dbpedia.org/ontology/visitorsTotal)
(http://dbpedia.org/ontology/architecturalStyle)
(http://dbpedia.org/ontology/nearestCity)
(http://dbpedia.org/ontology/architectualBureau)
(http://dbpedia.org/ontology/floorArea)
(http://dbpedia.org/ontology/rebuildingDate)
(http://dbpedia.org/ontology/areaLand)
(http://dbpedia.org/ontology/areaTotal)
(http://dbpedia.org/ontology/cost)
(http://dbpedia.org/ontology/elevation)
(http://dbpedia.org/ontology/awards)
(http://dbpedia.org/ontology/minimumElevation)
(http://dbpedia.org/ontology/maximumElevation)
(http://dbpedia.org/ontology/architect)
(http://dbpedia.org/ontology/timeZone)
(http://dbpedia.org/ontology/daylightSavingTimeZone)
(http://dbpedia.org/ontology/annualTemperature)
(http://dbpedia.org/ontology/Building/floorArea)
(http://dbpedia.org/ontology/tenant)
(http://dbpedia.org/ontology/areaWater)
```

There are multiple geospatial ontologies available on the Web. One of them is the DBPedia ontology which contains basic geographical classes and properties. Therefore, it was decided to

use Geonames feature codes and descriptions in order to integrate more classes and features to the main DBPedia ontology. This is because the DBPedia ontology is an upper level one and therefore lacks many geographical classes and their connections with other super and sub classes.

Moreover, it was found there was not one geospatial comprehensive ontology that covered every concept and relation needed for the data management of the GIS systems. A geographic ontology includes geo-feature classes and concepts, and geographic relations (e.g. topological, distance, and direction relationships) [HR04]. We cover the geo classes through integrating the DBPedia ontology with new directory ontology, and the geographic relations were covered through the creation of our relations ontology.

Two steps were involved in the whole integration process, building concepts taxonomy and relationships specifications. A concepts taxonomy is known to be the Directory Ontology [ZLT+08].

6.2.2 Building The Directory Ontology:

This step aimed at building the hierarchical structure for all the geographical entities identified in Geonames. The hierarchical structure basically refers to the class-subclass relationships between the different geographical concepts. The main steps of this process are summarized below:

- 614 feature codes were downloaded from Geonames [Geo10].
- The description of each code was analyzed and further subclasses where identified and split into subclasses of that code. Furthermore, classes that were identified to be super classes but do not have a feature code in Geonames were created and assigned a geo-code.

For example consider the below features codes with their descriptions:

Table 6-2: An example of Geonames feature codes with their descriptions.

Feature code	Description
populated place	a city, town, village, or other agglomeration of buildings where people live and work
medical center	a complex of health care buildings including two or more of the following: hospital, medical school, clinic, pharmacy, doctor's offices, etc.

The populated place was identified to be a super-class for city, town, village and building because all of the subclasses hold the "Is A" relationship with populated place. Next, the medical center was identified to be a health care building but there was not such a feature in Geonames, therefore a new feature code was created and named, "*Health care building*". Then, the new feature code was included under the closest super-class which was in this case "building," and so on. The hierarchy representation of the above example can be expressed as follows:

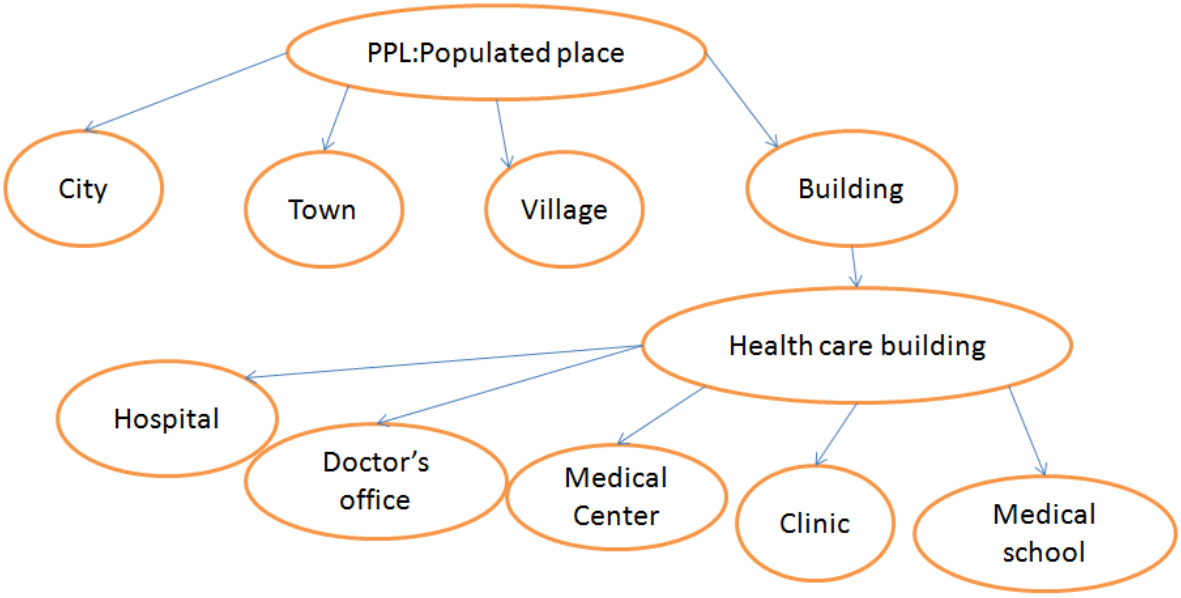

Figure 6-2: A small subset of the created ontology hierarchy. The source of the ontology is the Geonames geographical codes and descriptions.

The final result of this analysis is a text file that lists each class with its super class next to it separated by a separator ":" as shown below:

City: Populated place

Town: Populated place

Village: Populated place

Building: Populated place

Health Care Building: Building

Hospital: Health Care Building

Doctor's office: Health Care Building

Medical Center: Health Care Building

Clinic: Health Care Building

Medical School: Health Care Building

- A program was developed to process the text file and create the class hierarchy in Protégé [Pro03] using the Protégé OWL API [KFNM04] including the description of each class if it existed in Geonames. The resulting directory ontology is shown below.

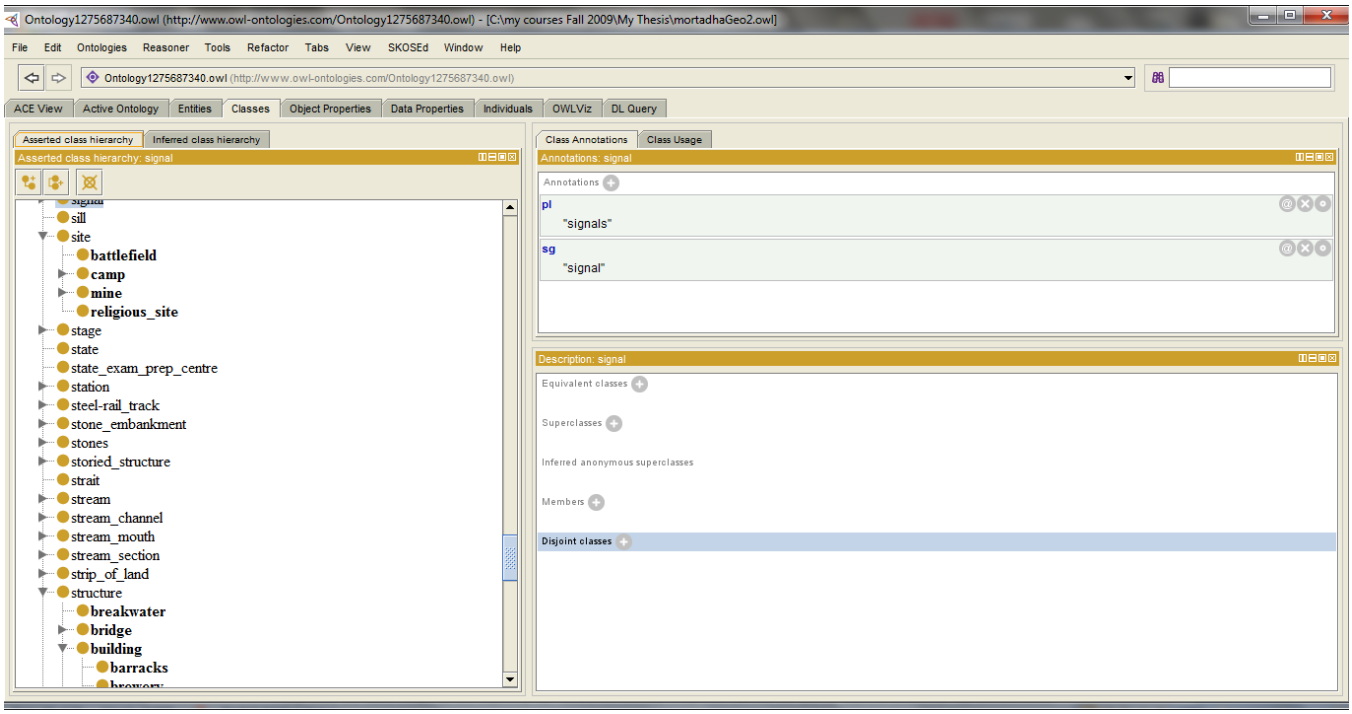

Figure 6-3: The complete generated directory ontology displayed in Protégé.

6.2.3 Relationships Specifications:

The aim of this step was to define an approximate specification of some geographical relationships. The different geographical relationships were extracted from different ontologies that exist on the Web. Each relationship was compared against others to discover the equivalences, inverses, children or parents of that relationship. Furthermore, each relationship was examined and specified whether it satisfied the logical properties like functional, transitive,

symmetric, asymmetric, reflexive, irreflexive and inverse functional. Table 6-3 shows a subset of the common geographical relations with their equivalences and inverses. Table 6-4 shows some properties of the relations. We have defined those properties to show how they can be used in reasoning which will be explained later.

- Functional property: A property is said to be functional if it relates an individual to only one individual.
- Inverse Functional: A property, P, is said to be inverse functional if the inverse property of P is also functional.
- Transitive: If individual A is related to B with some property P, B is also related to C with P implies A is related to C with P, then P is a transitive property.
- Symmetric: When A is related to B with a property P, and B is also related to A with the same property P, the P is a symmetric property.
- Asymmetric: If A is related to B with some property P and therefore is impossible to be related to A with P, then P is asymmetric.
- Reflexive: Is a property P relates A to some individual B and if A must also relate with itself, then P is reflexive.
- Irreflexive: A property P is irreflexive when A is related to B and therefore A and B are not the same.

Table 6-3: Geographical relations with their equivalences and inverses.

Relation	Equivalences	Inverses
in	locatedIn containedIn	Contains

	within	
Nearby	Nearby closeTo closeBy near	
neighborTo	Around nextTo	
at	By	
southOf		northOf
northOf		southOf
eastOf		westOf
westOf		eastOf
northEastOf		southWestOf
northWestOf		southEastOf
southEastOf		northWestOf
southWestOf		north EastOf

Table 6-4: Properties of geographical relations.

Note: Equivalent relations follow the same approach.

Relation	Functional	Transitive	Symmetric	Asymmetric	Reflexive	Irreflixive	Inverse functional
in		Yes		Yes		Yes	
Nearby			Yes			Yes	
neighborTo			Yes			Yes	
at		Yes	Yes		Yes		
southOf		Yes		Yes		Yes	
northOf		Yes		Yes		Yes	
eastOf		Yes		Yes		Yes	
westOf		Yes		Yes		Yes	

north EastOf		Yes		Yes		Yes	
northWestOf		Yes		Yes		Yes	
southEastOf		Yes		Yes		Yes	
southWestof		Yes		Yes		Yes	

In order to illustrate how these relationships and their specifications contributed to the geospatial semantics of the data for a powerful reasoning task, a simple ontology was used. The DBPedia ontology could not be used for this purpose at the moment because it did not define any complex geospatial relationships neither in its T-Box nor in the A-Box. Even if it was possible to find a comprehensive geo-ontology which defined the geo-relations with their properties between the classes, missing these important geo-relations between the A-Box instances of DBPedia did not enable us connecting the retrieved instance and reasoning over them.

The simplified ontology assumed a subset of a geographic A-BOX that was populated starting from a T-Box ontology. This enabled the search system not only to reason over the possible and inferred concepts, but to also discover and reason over the geo-relationships among the different instances. First, a subset of the simple ontology is introduced and second an example is presented in order to clarify the points. The simple ontology consists of the class structure in figure 6-4:

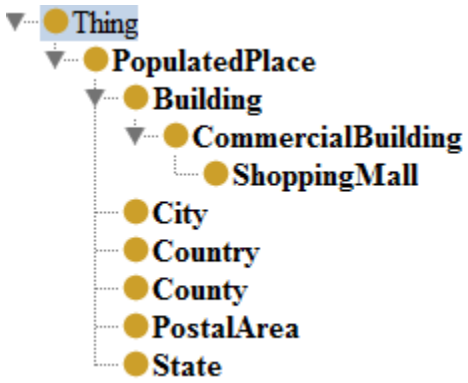

Figure 6-4: Simple ontology class structure

The following individuals were also created in the system and linked using the geographical relations defined in table 6-3 and table 6-4:

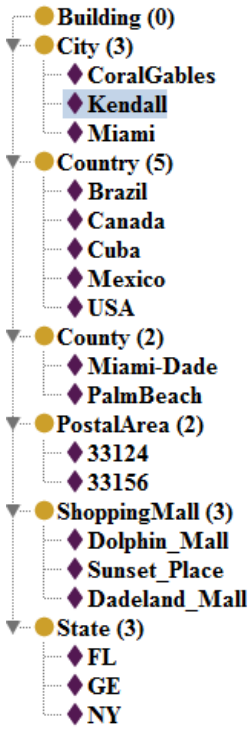

Figure 6-5: Simple ontology instance structure

- Example 1: This example shows how the reasoning system was able to discover implicit relations from the defined properties without explicitly defining these relations between the instances. Assuming the user was searching for "*Commercial Buildings located north of Kendall*". In the simple ontology the following facts were explicitly added:
    - Miami is located south of Miami-Dade
    - Kendall is located south of Miami
    - Dolphin Mall is located in Miami-Dade

Transforming the above user query into a one that is understandable by the Pellet reasoner yields the following : "*CommercialBuilding* and *PartOf* some *PopulatedPlace* and *northOf* value *Kendall*". The reasoner reported "*Dolphin Mall*" as a result of the query as shown in figure 6-5, although it was not explicitly specified that the mall was located north or south of Miami-Dade. Instead, it was only specified that it was located in Miami-Dade. The reasoner conducted an examination process through which it discovered the properties of the relationships and found that the given relation "*southOf*" is an inverse of "*northOf*" and both relations are symmetric. Therefore, the reasoner could end a result that would not be possible without the relation specifications defined above. The following series of facts were followed by the reasoner:

- Miami is a populated place located north of Kendall
- Miami-Dade is a populated place and is north of Miami city
- Dolphin Mall is part of Miami-Dade
- Dolphin Mall is a commercial building.

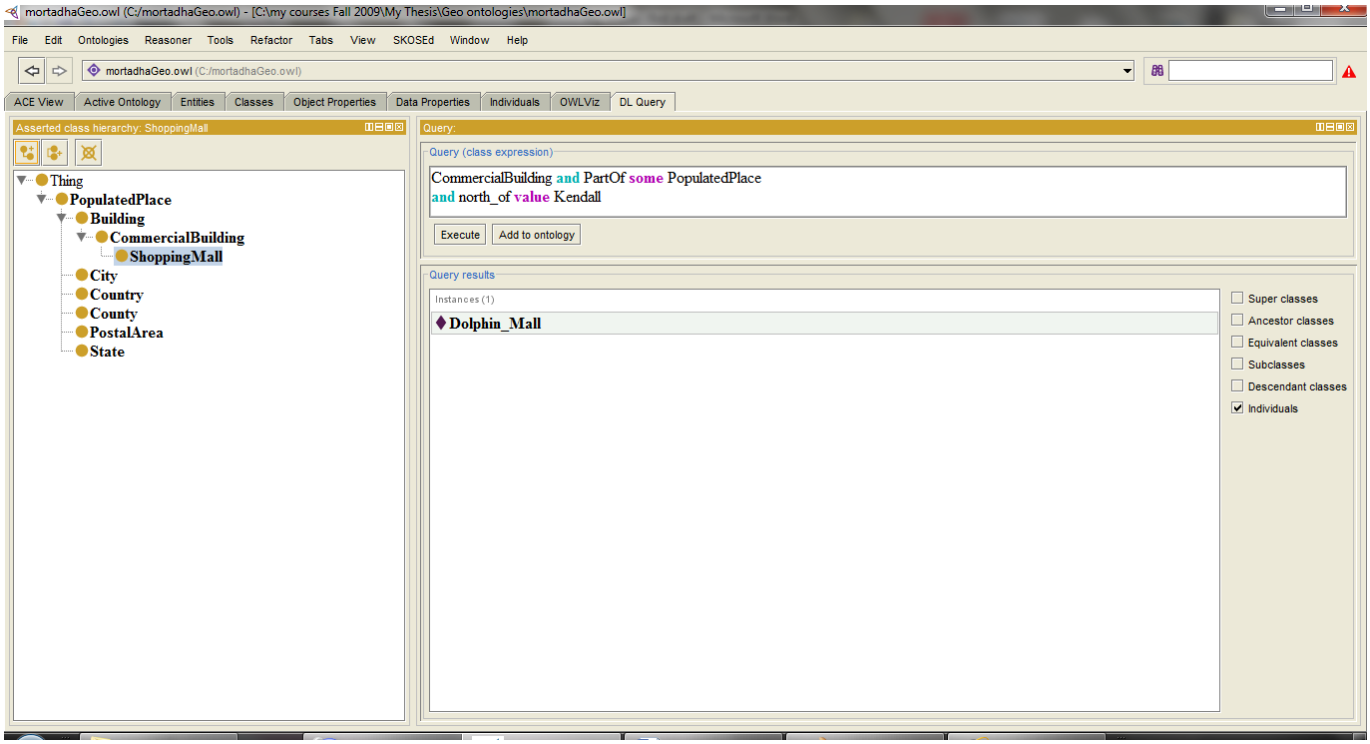

Figure 6-6: Querying the simple geo ontology using DL Query tool of protégé.

6.3 Geospatial Resources Discovery:

DBPedia was selected [ABK+08] as a searching knowledge base from to retrieve the resources requested by the user and to apply on them the semantic search techniques that were developed. One of the reasons for this selection was because DBpedia plays the role of a central interlinking hub, this means if we are able to connect to DBpedia, then we are also connected to data sources such as Geonames, the World Factbook, UMBEL, EuroStat, YAGO [ALH09] and more. As of April 2010, there are more than 4.9 million interlinks between DBpedia and external datasets including: Freebase, OpenCyc, UMBEL, GeoNames, Musicbrainz, CIA World Fact Book, DBLP, Project Gutenberg, DBtune Jamendo, Eurostat, Uniprot, Bio2RDF, and US Census data [Wik10]. Therefore, the richness and the interlinking capability to other data sources makes

DBPedia a good candidate to achieve one of the purposes such as adding a richer dimension to TerraFly by integrating it with other semantic knowledge bases.

However, the DBPedia resources were found to be missing critical geographical properties. For example, two towers in China were represented by two resources in DBpedia and both resources had similar but also different properties. Therefore, one tower was geotagged or assigned a latitude property which indicated that it was a geographical entity but the other tower did not. This motivated us to develop a mechanism to deal with such missing data especially if the result of the search was the best one that meets the user query but was not recognized to be a geographical entity.

6.3.1 Analysis:

A query was run to retrieve about 200 geographical entities in DBPedia, given their names. Then, it was discovered that about 23% of these entities were not identified as geographical entities simply due to the missing geographical properties as in the resource *http://dbpedia.org/resource/Miami_Sunset_High_School.* Although this entity connects to State and City as properties, it cannot be judged that this was a geo-entity because these two properties could also connect to events like theater shows.

6.3.2 Geographical Property Learning:

The steps below were conducted in order to come up with a heuristic methodology to differentiate the geographical resources from others:

- A program was developed and run to retrieve 1370 geographical resources from DBPedia. Only resources with the property (http://www.georss.org/georss/point) were considered geographical entities because it was believed it was a unique geo-property. The point property combines information about the latitude and longitude. Then, Jena API was used to extract all the properties from all the resources and to save them in a file. A total of 134,789 properties were extracted.
- Duplicates were then filtered in order to end up with a total of about 2898 unique properties as a mixture of geo and non-geo properties.
- The non-geo properties were manually removed by identifying them one-by one. Finally, a total of 462 properties were obtained.
- The geo properties were manually ranked starting from the most common identifier of geo-entities and ending up with the least identifier as shown in table 6-5. This was important for the semantic search service in terms of accuracy and efficiency. Recently, a query was run to calculate all the frequencies for all our extracted geo-properties. Each frequency number represented how many resources of a property appeared in the entire knowledge base. The property with rank number 1 was the latitude property (*http://www.w3.org/2003/01/geo/wgs84_pos#lat*). To improve the efficiency, the program will first validate if the latitude property has a connection with the current resource and if does, it will discontinue checking other connections. However, if the resource does not connect to the latitude property, the program will continue to validate other connections until it find one.

Table 6-5: The top part of the geo properties representing the highest rankings and the most common.

Property	Frequency
http://www.w3.org/2003/01/geo/wgs84_pos#lat	652,451
http://www.w3.org/2003/01/geo/wgs84_pos#long	651,890
http://www.w3.org/2003/01/geo/wgs84_pos#geometry	577,034
http://www.georss.org/georss/point	543,943
http://dbpedia.org/ontology/populationTotal	198,807
http://dbpedia.org/ontology/location	83,673
http://dbpedia.org/ontology/location	83,673
http://dbpedia.org/ontology/postalCode	83,482
http://dbpedia.org/ontology/populationAsOf	17,876
http://dbpedia.org/property/population	14,044
http://dbpedia.org/property/area	9,132
http://dbpedia.org/ontology/locationCountry	6,606
http://dbpedia.org/ontology/populationPlace	2,768
http://dbpedia.org/property/address	1,789
http://dbpedia.org/property/alt	1,047
http://dbpedia.org/property/areaAlt	1,018
http://dbpedia.org/property/altitude	519

SPARQL offers the "OPTIONAL" operator which can be used in order to dynamically check the existence of the geographical tags in a found resource. The optional operator is used to select data that is linked to a predicate when the predicate appears in several different terms. Each term might be used in a different subset of resources. For example, the below query will use the OPTIONAL operator to identify the geo-resources based on placing more than one condition. If one condition is satisfied the resource would be selected.

Select ?uri ?geoTag where{

OPTIONAL *{?uri <http://www.w3.org/2003/01/geo/wgs84_pos#lat > ?geoTag}*

OPTIONAL *{?uri <http://www.w3.org/2003/01/geo/wgs84_pos#long > ?geoTag}*

OPTIONAL *{?uri < http://www.w3.org/2003/01/geo/wgs84_pos#geometry > ?geoTag}*

OPTIONAL *{?uri <http://www.georss.org/georss/point> ?geoTag}*

OPTIONAL *{?uri < http://dbpedia.org/ontology/location > ?geoTag}*

}

However, many OPTIONAL operators can increase the computation cost because of many optional checks the SPARQL processor has to perform. Therefore, only a subset of the above tags was selected to identify the geo-relations, specifically the first five properties from table 6-5.

6.4 Query Design:

The query design model consisted of two parts. First, the direct search through which a direct query will be executed without any semantic evaluation of the query meanings. Second, the semantic query was formed using the semantic parsing, interpretation, execution and ranking. The reason for this division was to enhance the performance and to prevent against increasing the processing cost while obtaining the target results easily in case the user entered a direct resource name.

6.4.1 Direct Query Design:

Direct query considers evaluating the resources' labels if they include the key terms of the user query. Assume the query key terms are represented by the set T = $\{t_1, t_2, t_3 \ldots t_n\}$ and a candidate document C is under evaluation whether it meets the conditions to be included in the result set R. Given that C has a label L with a set of terms $CL_t = \{cl_1, cl_2 \ldots cl_m\}$. Then, $C \in R$ iff $T \subset CL_t$. The formal representation of the SPARQL query can be written as:

SELECT DISTINCT ?uri ?label ?abstract ?point WHERE {

?uri rdfs:label ?label.

?label <bif:contains> t_1 *AND* t_2 *AND* t_3 *... AND* t_n.

?uri rdfs:comment ?abstract.

OPTIONAL {?uri <http://www.georss.org/georss/point> ?point.

}

The ranking of the results returned from this query is based on Jaro Distance that is calculated for each result. The Jaro distance is used here to measure how well the words of the result label match the user query represented by the terms T. The Jaro distance will be given a detailed analysis in the coming sections.

6.4.2 Semantic Query Design:

The semantic query design process has taken an iterative model through which several combinations of the query unions were tested. A query union is a processing block that contains its own selection conditions and RDF triples which are evaluated by the SPARQL processor in order to retrieve the satisfying documents from the knowledge base. The SPARQL processor then combines all the resources resulting from the unions in order to constitute the final query result set. In order to retrieve a comprehensive list of resources that satisfies the user query semantics two lists of requirements were designed for the search system in an attempt to achieve a balance between the two sets as they are in contrast to each other. The first set consists of the performance requirements which include:

- **Response time**: This is the time the system takes in order to finish the search process.
- **Number of retrieved resources**: This represents the capacity as the maximum number of resources the system is able to retrieve from the search server to the client. Currently, the DBPedia query processor is able to retrieve a maximum of 1,000 resources at a time. However, this number can be much less if the maximum response time has been reached.
- **Full query evaluation**: This represents the system ability to evaluate all the query unions fairly. It means the system should not retrieve the maximum number of resources just by evaluating one query union but also by distributing the evaluation process among all the unions.

Another list that constitutes the comprehensiveness requirements includes:

- **Internal Structure Data**: This represents a number of properties that are required to evaluate the resources and rank them accordingly. As retrieving all the properties is an

expensive process that can greatly reduce the recall as we mentioned earlier, only a subset of properties are included as a tradeoff between performance and Comprehensiveness.

- **Geo resources identification data**: This represents evaluating a predefined set of tags of a resource dynamically in order to decide if it is a geographical resource, thereby including it as part of the result set. Since this process reduces the efficiency of the system, an evaluation process was also applied where only the most important tags were compared.
- **Evaluation of the geospatial coordinates:** This is responsible for a dynamic check if a resource is within the limited geo-area calculated before processing the query. This geo-area represents a bounding box around the location specified by the user in the query.
- **Evaluating resource with labels that meet the place semantics:** This evaluation can produce a large number of resources. It occurred that the SPARQL endpoint will sometimes retrieve all the resources from this part.
- **User query keywords evaluation:** Some resources are better retrieved by evaluating their abstract or description rather than evaluating their labels. Such resources can be fetched by matching the user query keywords with words of the resource abstract. For example, a resource with a label describing the resource owner name is not of much help. Instead, we can better search for the resource type, location, description and more through its abstract.
- **Evaluating resources with labels that meet the exact phrase of the user query:** A user might have an exact place name that does not contain any more description or geo-

information. This kind of query should be evaluated separately without considering other semantics because of a specific user request.

Since the launch of this research, more than 100 queries have been processed each of which constituted a combination of several query unions. Out of these experiments the following results were noted:

Table 6-6: Performance versus comprehensiveness requirements.

Requirement	Versus
Internal Structure Data	Response time, Number of retrieved resources
Geo resources identification data	Response time
Evaluation of the geospatial coordinates	Response time
Place semantics evaluation	Full query evaluation
User query keywords evaluation	Response time
User query exact phrase evaluation	Response time

The final iterative model consists of the query unions as shown in table 6-7.

Table 6-7: Final SPARQL unions that constitute SGSS query.

Requirement	Query	Notes
Internal structure Data and place semantic evaluation	Select `?uri ?label ?abstract ?point ?type ?property1 property2 Where ?uri rdfs:label ?label. ?uri rdfs:comment ?abstract. ?label <bif:contains> placePhrases. ?uri ISProperty1 property1.`	Propert1 and property2 are just examples of matching resource properties with those of the ontology concept. The number of properties can be more than two of course. placePhrases are semantic terms of

	?uri ISProperty2 *property2*. ?uri geo:point ?point. ?uri geo:lat ?lat . ?uri geo:long ?lon.	the query terms.
Geo resources identification data	OPTIONAL {?uri geo:point ?point.} OPTIONAL {?uri geo:lat ?lat.} OPTIONAL {?uri geo:long ?lon.}	This part Is applied to all the query unions
Evaluation of the geospatial coordinates	Filter (?lat <= latMax && // maximum lat ?lon >= lonMin && // minimum long ?lat >=latMin && // min latitude ?lon <= lonMax) // maximu	
User query keywords evaluation	Select ?uri ?label ?abstract ?point Where ?uri rdfs:label ?label. ?uri rdfs:comment ?abstract. ?abstract <bif:contains> QueryKeywords	QueryKeywords are those keywords except the geographical relations.
User query exact phrase evaluation	Select ?uri ?label ?abstract ?point ?property Where ?uri rdfs:label ?label. ?label <bif:contains> UserQuery. ?uri rdfs:comment ?abstract. OPTIONAL {?uri rdf:type <http://dbpedia.org/ontology/Place>.	

6.5 Semantic Ranking:

In the following section, the concept behind the semantic ranking for the semantic search results is detailed.

6.5.1 Concept:

The ranking used gives a better rank for results that best meet the user query semantics. One will argue that a user query might have many semantics and all of them are eligible to be ranked first. The approach used here ranks the semantic results according to a total score that can split and rank them even if they have close meanings. It depends on several evaluation scores that are calculated for each result and then combined to represent the overall score for that result. The different ranking scores takes into account the internal structure of the resource, the semantic matches between the resource label and the user query terms and the spatial distance between the location specified by the user and the location identified by the latitude and longitude of the resource. The ranking problem is formalized in the next section, followed by the string matching function. Then, each score calculation is detailed followed by the combination process with the overall ranking algorithm. Finally, a cost analysis of the ranking algorithm is presented.

6.5.2 Ranking Problem:

Assume we are searching an open knowledge base (KB) for the best results that meet the user geographical query *Q*, which has the following geospatial query structure: *Place-Relation-Site* where the site is either a specific location e.g "*North Miami Beach*" or another place e.g. "*Ocean Drive*". For simplicity, the assumptions are listed as follows:

- Q_{sp} is the set of semantic terms for the query place name specified by the user.
- R = $\{r_1, r_2, r_3 \ldots r_n\}$ is the set of the initial results retrieved from KB.
- r_j L is the label of the result r_j
- $r_j L_k$ is the term k of the label r_j L that belongs to the resource r_j

- r_j P is a subset of properties that were retrieved for the result r_j
- $OntP$ is the set of properties extracted from the underlying ontology. Those properties are extracted from the concept of the place term identified from the user query.
- Q_{lat} is the latitude value of the query location specified by the user query
- Q_{lng} is the longitude value of the query location specified by the user query
- r_j Lat is the latitude value specified in the resource r_j
- r_j Lng is the longitude value specified in the resource r_j

The problem is to define the ranking function Rank(r_j) that assigns a ranking number for the resource r_j given all the above inputs. The Rank function should assign the smallest value to the most relevant result that best meets the user query or one of its semantics. The smaller the rank number assigned to a resource the more relevant that resource is.

6.5.3 Matching Function:

The purpose of the matching function is to calculate if two strings are matched. This function represents our basis in deciding if two properties or two terms are matched or not. Knowing if two terms are matched or almost matched is very beneficial especially if we want to compare two properties and they are the same but they were annotated differently e.g. *Location* and *locatedIn*. However, we have limited the usage of these matching scores in order to improve the ranking time.

The basis for the matching between terms is to use the Winkler [Win99] algorithm, which is an improvement of the Jaro distance algorithm [Jar89] by adding extra weight to common prefixes. The Jaro algorithm counts the number of common characters in two strings if they are within half the length of the shortest string of its position in the other string. The algorithm records the number of these common characters which occur in order and the number of those characters occurring out of order [TDG07]. The Jaro distance between two strings s_1 and s_2 can be computed as follows [Jar89]:

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)$$

Where m is the number of matching characters. Two characters are considered matching only if they are not more than

$$\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1$$

where t is the number of transpositions.

Winkler distance [Win99] is defined as below:

$$d_w = d_j + (l.p(1 - d_j))$$

Where the d_j is the Jaro distance defined above, p is a scale factor and is usually given the value of 0.1 and l is the length of common prefix at the start of the string with a maximum of 4 characters.

The matching function as shown in the equation below can then be calculated based on Jaro-Winkler distance. Match(i, j) measures the match between two strings *i* and *j*. It outputs 1 (means matched) if the Jaro-Winkler is greater than or equals to a threshhold value and 0 (not matched) otherwise. The threshold value was changed until the best results emerged approximately 0.8. The best results are those terms which were successfully matched with their similar terms and the similar terms carry the same meaning.

$$Match(i,j) = \begin{cases} 1 & if\ JaroWink(i,j) \geq Threshhold \\ 0 & if\ JaroWink(i,j) < Threshhold \end{cases}$$

6.5.4 Ranking Scores:

The semantic ranking considers 3 scores as follows:

- **Score of document internal structure (*factor1Score*)**: The basic idea of this score is to measure how well the resource r_j in representing the ontology concept represented by its property set *OntP*. Because of scalability and performance issues with the DBpedia query response, retrieving the internal structure of every resource drastically reduces recall as shown in figure 6-6, and therefore is not practical. Alternatively, we only retrieve a subset of a resource property specifically when there is a match with the data type properties of the ontology concept. This can be illustrated in an example. Assume there is a query word *W* for which there is a desire to retrieve the instances from our underlying knowledge base. *W* is represented by an ontology concept *C* which consists of a set of properties *OntP*. $OntP_{obj}$ and $OntP_{data}$ are the object and data type properties of C. That is

$$OntP = OntP_{obj} \cup OntP_{data}$$

Instead of retrieving and matching *OntP* against all the properties of a resource r_j, we only match $OntP_{data}$ against the data type properties of r_j. From now on, we will refer to $OntP_{data}$ as $OntP$ for simplicity.

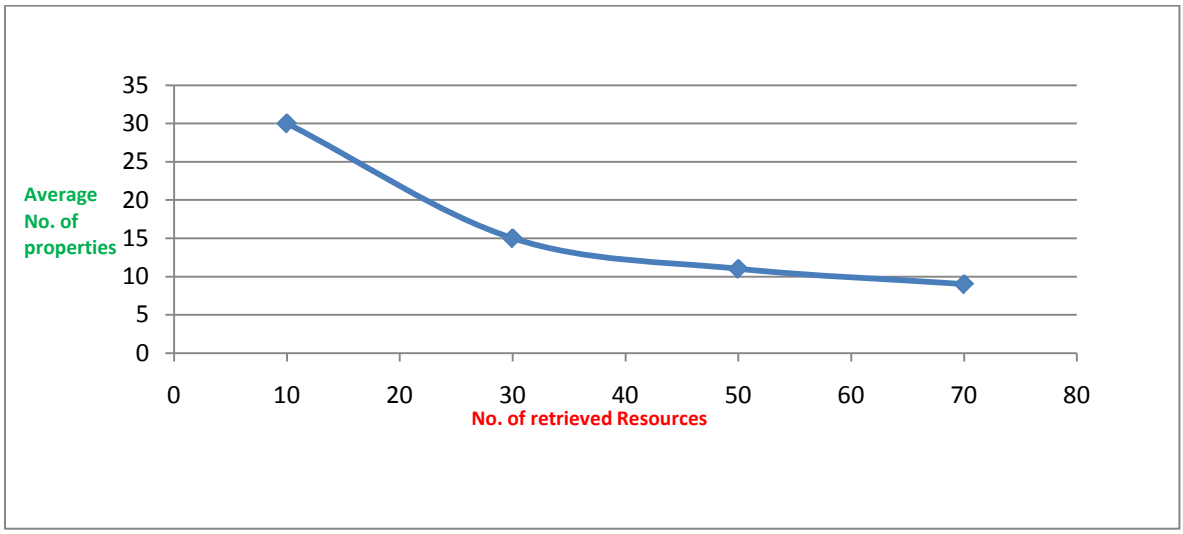

Figure 6-7: A tradeoff between number of retrieved resources with average number of properties in all the resources.

In figure 6-6, we notice that there is a tradeoff between number of retrieved resources and average number of properties per resource. Since recall is proportional to the number of retrieved resources, recall increases when the number of retrieved properties is reduced. Therefore, we retrieve only a subset of the resource properties in order to increase recall.

The internal structure score is measured by the following equation:

$$Factor_1Score = \frac{\sum_{k=1}^{N} Match(r_j P_k, OntP_c)}{|OntP|}$$

Where N = $|r_j P|$ and $r_j P_k$ is the property k of the set $r_j P$

$OntP_c$ is the closest property from the set *OntP* to the property $r_j P_k$

This equation sums the number of matched properties from the resource r_j with the properties in *OntP*. For example, if the set $r_j P$ = *{name, label, location, zip code, latitude, architect}* and the set *OntP= {name, label, structure, locatedIn, zip code}*, then *factor1score* for the resource r_j will compare the above properties as shown in table 6-8.

Table 6-8: An example for evaluating the internal structure matching of a resource with that of the concept represented by the ontology. The match value uses a string matching score based on Jaro-winkler algorithm.

Property $r_j P_k$	Closest property in OntP	Jaro-winkler	Match($r_j P_k$, OntPc)	Match?
Name	Name	1	1	Yes
Label	Label	1	1	Yes
Location	locatedIn	0.805	1	Yes
Zip code	Zip code	1	1	Yes
Structure	Architect	0.57	0	No

As a result, the *factor1Score* for r_j is 4. Later, you will see how this score is normalized and assigned a specific weight before it is added and combined to the total score in order to produce a relatively good rank.

- **Score of geographical place semantic match**: This score is very similar to the previous one in terms of comparison. Previously, the properties of the resource were matched with the properties of the ontology. Here, the semantic terms of the query place are matched against the terms found in the resource label. This is because it was noticed that labels of DBPedia resources carry precise and short description about their resources which makes them good candidates for both accuracy and efficiency. The equation of this score is expressed as below:

$$Factor_2Score = \frac{\sum_{k=1}^{M} Match(r_j L_k, Q_{sp_c})}{|Qsp|}$$

Where M is the number of terms in the label r_j L and Qsp_c is the closest match of the query semantic place terms to the label term $r_j L_k$. For example, let the user query be "*Sunset school in Miami*" and let the resource label $r_j L$ = "*Sunset academia*".

Table 6-9 measures this score:

Table 6-9: An example for matching the terms of the resource label with semantic terms of the query place term. The matching score is based on Jaro-wikler algorithm.

Label term $r_j L_k$	Closest term in Qsp (Qspc)	Jaro-winkler	Match($r_j L_k, \text{Q}_{\text{sp}_c}$)	Match?
Sunset	Sunset	1	1	Yes
Academia	Academy	0.86	1	Yes
	School			
	College			
	University			

	Sundown			

The semantic place match score =1.86. Similarly, we will show in a later stage how to combine this score with others to form the overall ranking score.

- **Location Spatial Distance Score**: this score assigns a better rank to resources with locations closest to the locations or places specified by the user. To express this score in a better way let us examine the below equation:

$$factor_3 Score = \frac{1}{ScaledSpatialDistance(SSD)}$$

$$SSD = \frac{HaversineDistance(Q_{lat}, Q_{lng}, r_{lat}, r_{lng})}{farthest - closest} + 1$$

Farthest = the farthest location obtained from a resource compared to other resources' locations.

Closest = the closest location obtained from a resource compared to other resources' locations.

The score is affected by SSD (scaled spatial distance) so the bigger SSD value is the smaller score will be given to the current resource. In other words, those resources with locations closer to the location specified by the user will be given better scores. The SSD value measures the spherical distance between the resource location and the location specified by the user, but scaled to a smaller value in order to balance it with other scores. Haversine distance is one of the greatest circle distances which take the latitude and longitude values of two points on earth and measure the spherical distance between them. However, SSD distributes the scores fairly among the resources by considering the

distance between the farthest and closest locations found. As a result, a resource with a very far location from the user specified location but actually the closest one will not be given a very small score because it is still the closest.

6.5.5 Combination Process:

As we have seen earlier that each resource will be checked and examined in 4 dimensions, each corresponding to a score factor calculation. Now, we need to combine these scores in order to represent the overall ranking of the resource.

$$Rank(r_j) = \frac{1}{totalScore(r_j)}$$

Where,

$$totalScore(r_j) = \sum_{i=1}^{N} \alpha_i \times factor_{ij}Score$$

N is the number of factors which currently equals to 3.

$factor_{ij}Score$ is the score calculated for factor *i* for of the resource r_j and αi is the weight given to the factor *i* which is predetermined before the execution of the program and is fixed. These weights are under examination and evaluation and will be adjusted when we examine the user evaluation of different queries under different weights.

As can be seen from the rank equation, the bigger the calculated score for a given resource, the better the rank will be assigned to that resource.

6.5.6 Ranking Algorithm:

Procedure RANK

Inputs:

- User query Q
- OntP which is the set of properties extracted from the underlying ontology for the closest concept of the user entered place.
- r_j P is the set of properties found in the result r_j
- $R = \{r_1, r_2, r_3 \dots r_n\}$ which is the set of the initial results retrieved from KB.

Output:

R_{Ranked} is a set of objects representing the elements of R but ranked and ordered.

Begin

Define variables *factor1Score, factor2Score, factor3Score*.

factor1Score= factor2Score= factor3Score= factor4Score=0

Loop through *R*

$current\ r \rightarrow r_j$

/* calculating the first score*/

Loop through elements in $r_j P$

$current\ resource\ property \rightarrow r_j P_k$

Define variable *bestJaro*=0

Define variable $OntP_{closest}$

Loop through elements in $OntP$

$current\ ontology\ property \rightarrow OntP_{current}$

If jaro-wikler($r_j P_k$, $OntP_{current}$) > *bestJaro*

Then bestJaro = *jaro-wikler*($r_j P_k$, $OntP_{current}$) , $OntP_{closest} = OntP_{current}$

End loop

If *bestJaro > threshold*

Then *factor1Score* ++

End loop.

$factor1Score = factor1Score \ / \ |OntP|$

/*calculating the second score*/

Loop through elements in $r_j L$

$current\ term\ of\ the\ label \rightarrow r_j L_k$

Define variable *bestJaro=0*

Define variable $Q_{sp}{}_{closest}$

Loop through elements in *Qsp*

$current\ semantic\ place\ term \rightarrow Q_{sp}{}_{current}$

If *jaro-wikler(*$r_j L_k$, $Q_{sp}{}_{current}$ *) > bestJaro*

Then *bestJaro = jaro-wikler(*$r_j L_k$, $Q_{sp}{}_{current}$ *),* $Q_{sp}{}_{closest} = Q_{sp}{}_{current}$

End loop

If *bestJaro > threshold*

Then *factor2Score ++*

End loop.

$factor2Score = factor2Score \ / \ |Qsp|$

/* calculating the third score*/

/* calculate SSD (scaled Haversine distance) given earth points

$Factor3Score = \frac{1}{SPD\,(Q_{lat}, Q_{lng}, r_{jlat}, r_{jlng})}$

Define $r_j{}_{totalScore}$ = $\alpha_1 factor1Score + \alpha_2 factor2Score + \alpha_3 factor3Score$

Define an object $r_j Obj$

$r_j Obj.resource = r_j$

$r_j Obj.rank = \frac{1}{r_j{}_{totalScore}}$

Insert $r_j Obj$ into R_{Ranked}

End loop

Merge Sort all elements in R_{Ranked} .

Return R_{Ranked} .

End RANK.

6.5.7 Cost Analysis:

- Processing all resources costs N
- For each resource, the properties of the resource were compared with a subset of all the properties of the ontology and that costs $|r_j P| \times |OntP| = \beta$. In the worst case the $\beta = |OntP|^2$
- For each resource, the semantic place terms of the user query were compared with the label terms and in the worst case that costs $|r_j L| \times |Q_{sp}| = \gamma$.
- For each resource , Haversine distance was computed to cost 1
- Sorting costs $NlogN$ using the merge sort algorithm.

  Total cost = $(N \times |OntP|^2 \times \gamma) + NlogN$

  However, since the number of semantic place terms with the label terms of the resource is expected to be very small compared to other numbers, it can be considered as a constant.

  We can rewrite the cost equation above as:

  Cost = $(N \times |OntP|^2) + NlogN$

  ➔ Cost = N $(|OntP|^2 + logN)$

  However, in situations where the place concept has no ontology properties that is $|OntP| = 0$ the cost will be $N\ logN$

Chapter 7. SGSS Implementation

7.1 Infrastructure Development:

The infrastructure development consists of all the activities that are related to the installation, deployment; configuration, integration and learning of the different technologies and APIs that are needed to extract and process the semantic data. These technologies will be explained in details with our specific usages in the next section:

1- Protégé [Pro03]:

   Protégé is a free, open-source platform that provides tools to construct domain models and knowledge-based applications with ontologies [Pro03]. It supports the creation, visualization, and manipulation of ontologies in various representation formats. We used Protégé to manually manage our ontologies, create test individuals, and to run simple SPARQL queries on instances.

2- Protégé OWL API [Pro03]:

   It is an open Java library to create and maintain ontologies, add classes, properties, instances and maintain ontology attributes. It also provides capability to process SPARQL queries and to reason with OWL data models. We used this API for automatic integration between our ontologies.

3- Jena [HPDC09]:

Jena is a Java framework for developing Semantic Web applications. It provides a programmatic environment for RDF, RDFS, OWL, SPARQL and includes a rule-based inference engine [Jena]. It also supports reading and writing RDF in RDF/XML, N3 and N-Triples. Moreover, Jena has incorporated a SPARQL engine through which we can call an online knowledge base like DBPedia and pass it a query. Finally, Jena has built-in classes to format the query results returned from the queried knowledge base.

4- Protégé OWL Reasoner [Pro03]:

A built in reasoning API that can be used to access an external DIG compliant reasoner like Racer or Pellet, thereby enabling inferences to be made about classes and individuals in an ontology using OWL-DL. Inferred elements include but are not limited to inferred super classes, inferred equivalent classes, and inferred types for individuals. OWL-DL has its foundations in Description Logics, which are decidable fragments of First Order Logic. We are using this reasoner to discover inconsistencies between classes and individuals and to obtain inferred objects as well.

5- WordNet: Is a lexical database of English terms and concepts. It is used to get Synonyms, Hypernym, Hyponyms, MemberMeronyms and more forms of the word related concepts. We use WordNet in order to separate the user query terms into geo and non-geo terms. Moreover, we use WordNet as an extra step, besides the ontology processing, in order to capture meanings of the user query terms. WorldNet can be

used in a preprocessing step, before reasoning in the background ontology, in order to process only the terms identified as relevant to geography.

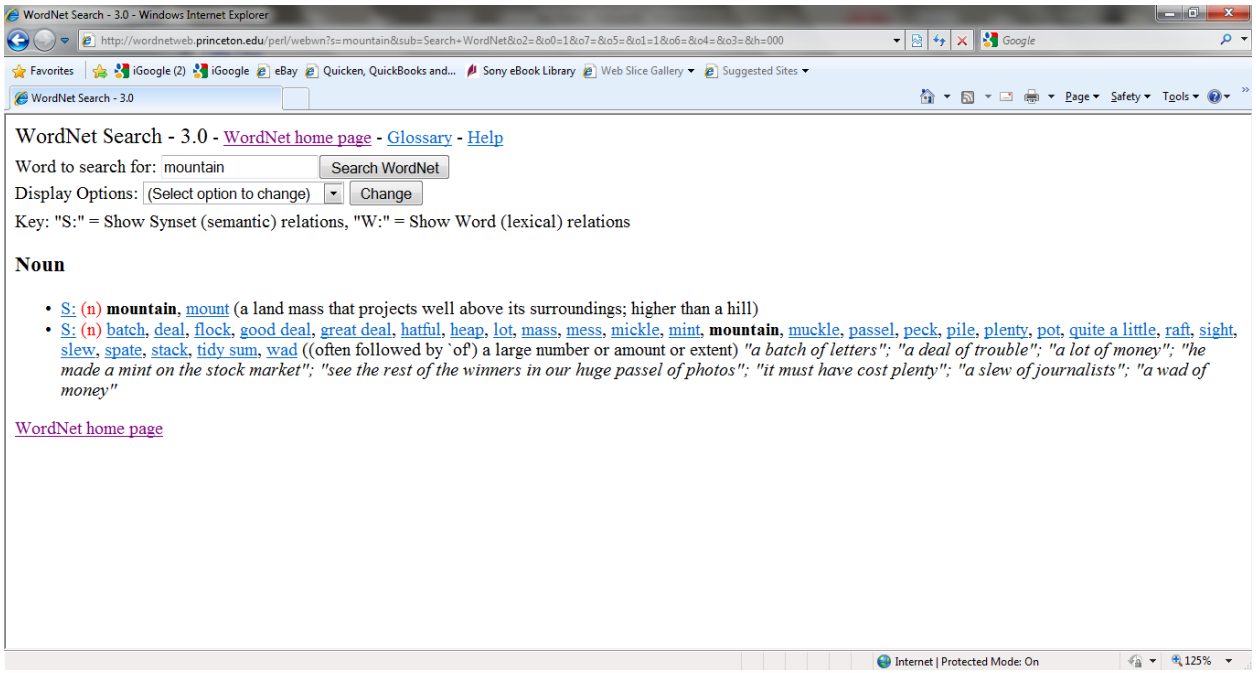

Figure 7-1: WordNet online access.

- **Synonym**:

  The synonym of a word is another form with different syntax of that word but carrying the same or similar meaning. For example "*car repair*" and "*auto maintenance*" can be considered two synonym phrases particularly "*car*" is a synonym of "*auto*" and "*repair*" is a synonym of "*maintenance*".

- **Hyponym**:

  Is used in natural language processing and is used to refer to a "*specialization*" relationship between two terms. We can assume that a hyponymy relationship

between two terms is very similar to the "*subClassOf*" relationship between classes in an ontology. For example, "*Green*" is one of the hyponym forms of "*color*".

- **Hypernym**:

  If a hyponymy is the specialization relationship between two terms, the hypernym is the generalization relation that can be considered the inverse relationship of the first one. The same example mentioned earlier "*color*" is the hypernym form of "*Green*".

6- DBPedia [ABK+08]: Is a project that aims at mapping Wikipedia resources to semantic Web data models including RDF and OWL models. DBPedia is using the infobox tables in Wikipedia pages to extract information, properties, linked resources, geographical coordinates and more in order to open another door towards the semantic Web linked data integration. As of April 2010, the DBpedia dataset describes more than 3.4 million things including 413,000 places [DBP10].

7- Geonames [Geo10]: Is a rich geographical database that contains over eight million geographical names and consists of 7 million unique features of 2.6 million populated places and 2.8 million alternate names. Moreover, Geonames is integrating geographical data such as names of places in various languages, elevation, population and others from various sources. All latitude and longitude coordinates are in WGS84 (World Geodetic System 1984). Our implementation of Geonames is for the future benefits of integrating more data from additional data sources using Web service.

8- SGSS Gazetteer: An extracted gazetteer from two geographical data sources namely GNIS and ADL gazetteers.

9- Suggester Spell check [LLC06]: A Java API providing recommendations for unknown words in user query for local search systems. A system administrator can create a list of preferred words and assign higher weight to such words. As a basic implementation Suggester can serve as a spellchecker. Based on fast edit-distance calculation algorithm enhanced with Lawrence Philips Metaphone algorithm and private fuzzy-matching algorithm. We use this API to correct wrong entered words in the search query text.

7.2 **System Architecture:**

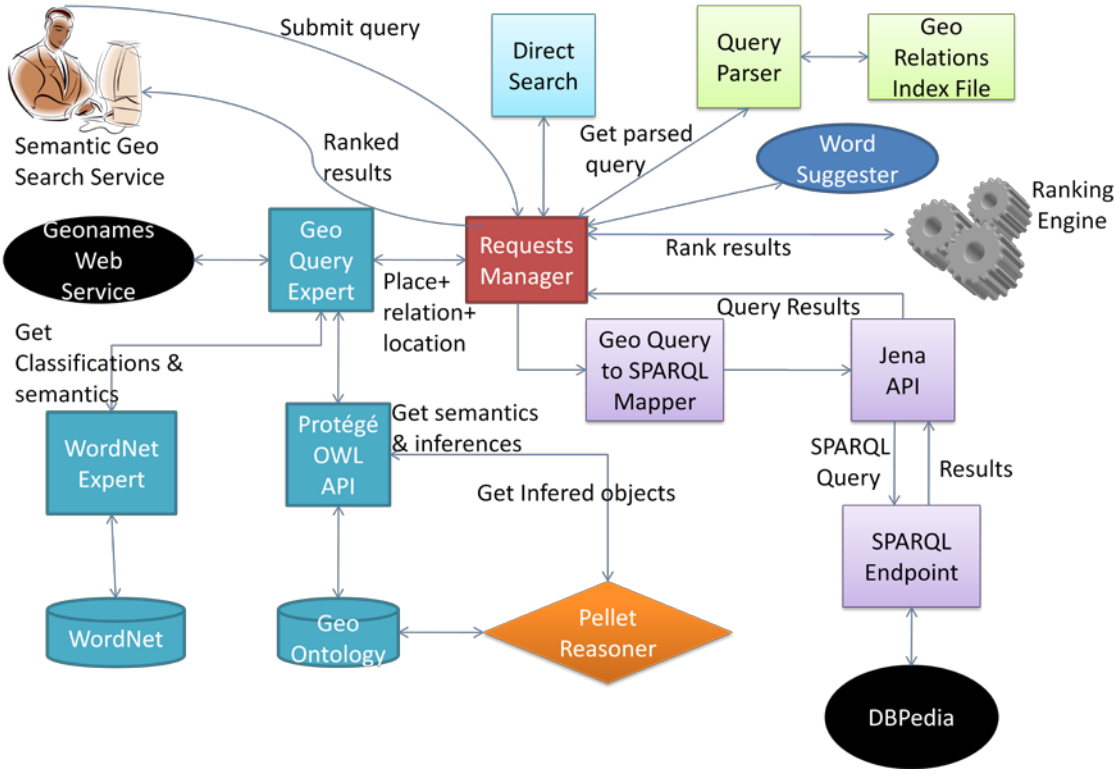

Figure 7-2: System Architecture of SGSS.

The following section explains the high level mechanism and communication processes between the different components of the semantic geo-search service. The task of each component, inputs, outputs and the issues associated with the task will be addressed. The components are explained in the same order they are executed in our program during run time.

- **Semantic Geo Search Service Client**:

  Users can access this service thorough a JSP page or a java client application that communicates locally with the different components on the same workstation and remotely with the knowledge bases like DBPedia and Geonames. Simply, the user enters a free text searching for a specific place adhering to the general geo-query structure which consists of (*Place, Relation and Site/Location)* and submits the query to the requests manager. Similarly, the client application waits for the results to be ranked and returned by the requests manager in order to be represented to the user.

- **Requests Manager**:

  The requests manager acts as a central processing unit with interfaces that communicate with almost all the components in the system. It receives the user request, formats it, interprets it, requests the search execution, requests ranking, and communicates back the ranked results to the user client application. Moreover, this component is responsible for logging all the errors and exceptions for better system monitoring.

- **Word Suggester**:

  Just like Google! Spelling mistakes and errors keyed by the user are taken care of through this component. The query text is passed to this suggester in order to check that every single word is according to the English dictionary. When there are errors, the suggester passes them back to the requests manager which will present them to the user if he wants to correct the query text.

- **Query Parser**:

  The parser searches a file that contains all the geographical relations in order to discover the exact location phrases. The parser will consider everything that lies before the relation as part of the place name and every word that follows the relation as part of the location name. However, smart parsing mechanism is required for more complicated user query which is out of the scope of this research.

- **Geo Query Expert (GQE):**

  The requests manager passes the identified place name, relation and location to the expert that will find the semantics of the terms and recompose them together again in order to create the geo-query object. In general, the geo-query object represents a package that must be submitted to the SPARQL mapper then to the knowledge base for processing the search. The geo-query object contains a subset of the place semantics, relation semantics according to the query context, and the inferred objects. First, GQE calls the WordNet expert in order to retrieve initial semantic terms. Then, the semantic terms are further processed through the Protégé OWL API that looks

into the ontology for super classes, subclasses, inferred classes and etc. forming a richer query semantics. Very general forms of super classes are removed from the semantic terms in order to narrow the search because they include irrelevant results. For example, the system considers "*building*" a semantic term from the query "*Restaurants in kansas city*" only because it found "*Building*" a super class for "*Restaurant*". Therefore, these types of general words like "*building*" and "*place*" are removed from the semantics list. Moreover, evaluating all the super classes, sub classes and lexical semantics affects the performance of the database query because it could extend to pages. For this reason we only consider a subset of all these sets. After that, the GQE interprets the relation semantics and specifications obtained from the ontology in order to define a distance along with the query location obtained from our SGSS gazetteer. Obtained place semantics and potential locations will be integrated into the geo-query object. The Geonames Web service provides a hierarchical navigation to any specific geographical entity in this world provided that we pass the Geonames id of a specific location. Therefore, first we obtain a list of potential locations that are closest to the location entered by the user and we select the first entity as we noticed that Geonames sorts the list beginning with the closest retrieved location name to the passed location name. Then, using the Geonames identifier we call another service to get the hierarchy of the intended location. For example, if the query location extracted from the user entered text was "Kendall", calling Geonames Web service provides us with the hierarchical structure shown in table 7-1.

Table 7-1: The above list shows an example of a list of Geonames hierarchical entities that are returned as an answer to searching for "*Kendall*".

ToponymName	Latitude	Longitude	Geonames ID	Entity Code
Earth	0.0	0.0	6295630	AREA
North America	46.0732306254083	-100.546875	6255149	CONT
United States	39.76	-98.5	6252001	PCLI
Florida	28.7505408	-82.5000976	4155751	ADM1
Miami-Dade County	25.6170505	-80.5331133	4164238	ADM2
Kendall	25.6792695	-80.3172743	4160711	PPL

For simplicity, and to narrow the results, we remove general entity names from the list like "*Earth*" "United states" and "*North America*" because they can enlarge our retrieved list and increase the computation cost. The right most column of the table indicates the type of entity as described in the Geonames ontology.

The geographical limit is the restricted geographical area within which we limit our search. In order to apply this restriction we need to define a circular area or a bounding box around our intended geographical point. We usually apply this kind of restriction by combining the coordinate's information obtained from our integrated SGSS gazetteer about the user location with the semantic of the geo relation entered by the user. The geo-relation gives an estimated radius to the restricted area. So, the combination of the coordinates with the radius produces a meaning of how far we should extend our search away from these coordinates. This limit can

be passed to the SPARQL endpoint as part of the query by specifying the limits of the latitudes and longitudes to be considered when searching the knowledge base. However, although this restriction produces an accuracy of more than 95% of the resources found in DBPedia, it has very important limitation. Passing such restriction requires all resources in DBPedia to have latitude and longitude points against which we can apply our restriction calculation. Unfortunately, many DBPedia resources are inconsistent and do not describe very critical geo-properties like the geo-coordinates due to incomplete geotagging. As a result, the query does not consider such entities and is not able to tell if they are within few miles away from the user location or they are in another planet, which reduces recall significantly. Figure 7-3 compares two approaches one with area restrictions and the other without. The tested queries are describing the search for different types of entities namely shopping malls, parks, hospitals and museums. Obviously, the approach without restrictions has bigger number of relevant results and therefore a better recall value. On the other hand, the one with geo restrictions reduces the computation complexity especially in ranking as the ranking complexity increases with the increase of the number of results.

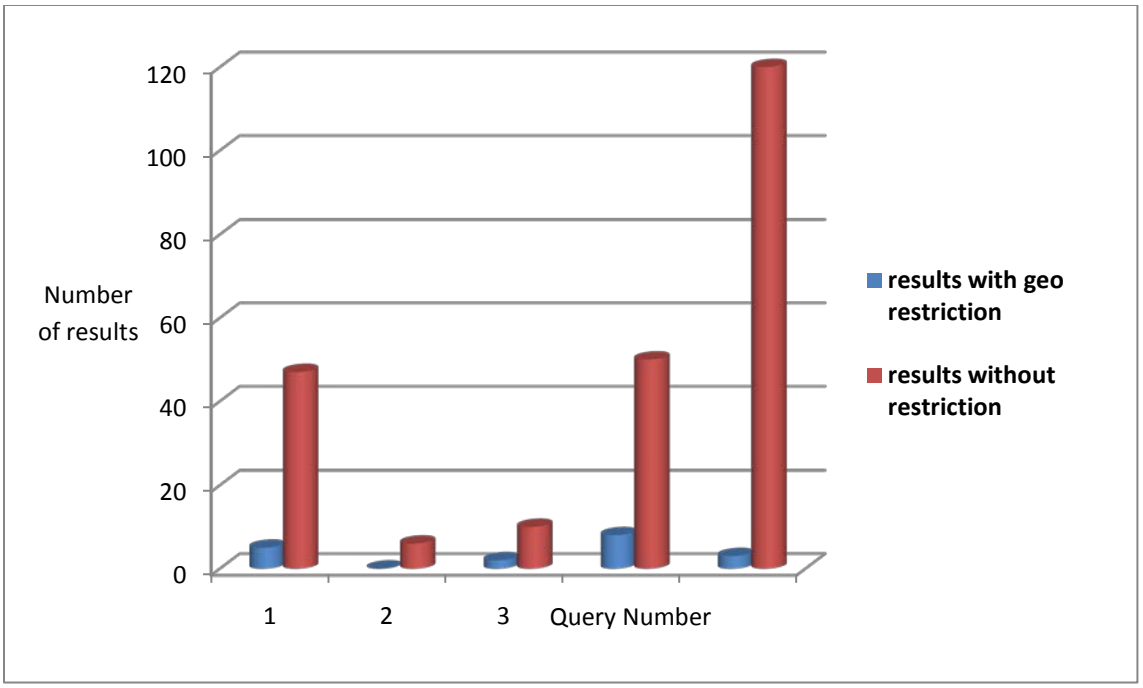

Figure 7-3: An illustration of 5 different scenarios each one was run in both approaches, with coordinates' restrictions and without.

- **WordNet Expert (WNE):**

  WNE combines a list of synonyms, hypernyms, hyponyms and memberMeronyms of the query keywords. As this list can be large that directly affects the efficiency of the query response, we limit this list of each term by considering only half the number of elements in the list in a random basis.

- **Protégé OWL API**:

  As we mentioned earlier, this API is a middleware between the program and the underlying ontology. Semantics of the concepts can be obtained by calling methods to

retrieve equivalent classes, subclasses, super classes, descriptions and to obtain alternatives through the *sameAs* property of the concept.

- **Pellet Reasoner**:

  We can instantiate whatever reasoner we want to use through the Protégé OWL API, provided that we have the reasoner installed on our workstation. Pellet is a Java written inference mechanism that manipulates description logic expressions and that is integrated with Jena API. Pellet was developed by the MindSwap Group at the University of Maryland [MG10]. In general, Pellet retrieves inferred objects like classes, subclasses, and inferred equivalent classes through its reasoning and inference engine and by using the underlying ontology rules.

- **Geo Query to SPARQL Query Mapper:**

  After obtaining the place semantics, user query semantics and potential locations to search, the geo-query object is considered ready for mapping and processing. Therefore, the requests manager passes the object to the mapper, which translates the information in the object into a low-level knowledge base SPARQL query.

- **Jena API:**

  Jena API checks the syntax of the translated SPARQL query and reports any errors back to the requester. This API is also used to set the connection parameters and preferences that must be communicated to the SPARQL endpoint in order to apply them on the search results. Parameters include the address of the Web service, and the

preferences include, the session timeout, number of results, results format like XML and so on.

- **DBPedia SPARQL Endpoint:**

  It is a conformant SPARQL protocol service that acts as an interface that was developed by DBPedia developers. It takes the SPARQL query as input, passes it to DBPedia query processors and returns the results back to the calling client.

- **Ranking Engine:**

  The ranking engine takes care of all the processes related to calculating the resource scores and combining them to form the final rank score based on the different features and properties of the resource. It also handles the sorting task according to the final rank value assigned to the resources.

7.3 User Interface:

The semantic geo service main interface provides a free text field for the user to enter a query, thereby reducing the rules that the user has to follow. As shown in the below interface, the suggester corrects any misspelled word and provides alternatives to the user to override the original query.

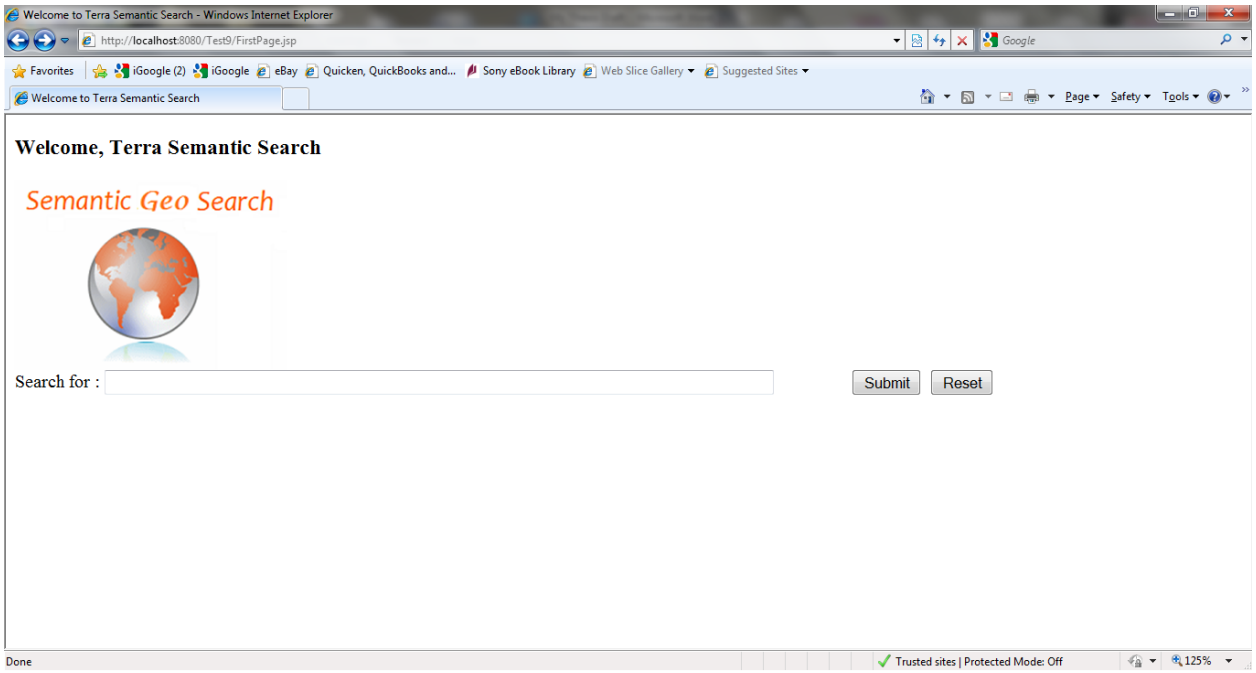

Figure 7-4: SGSS user interface.

The resulting view lists all the results ranked in ascending order so the first results are the best matches. Moreover, the resource URL, brief description, and location are also listed.

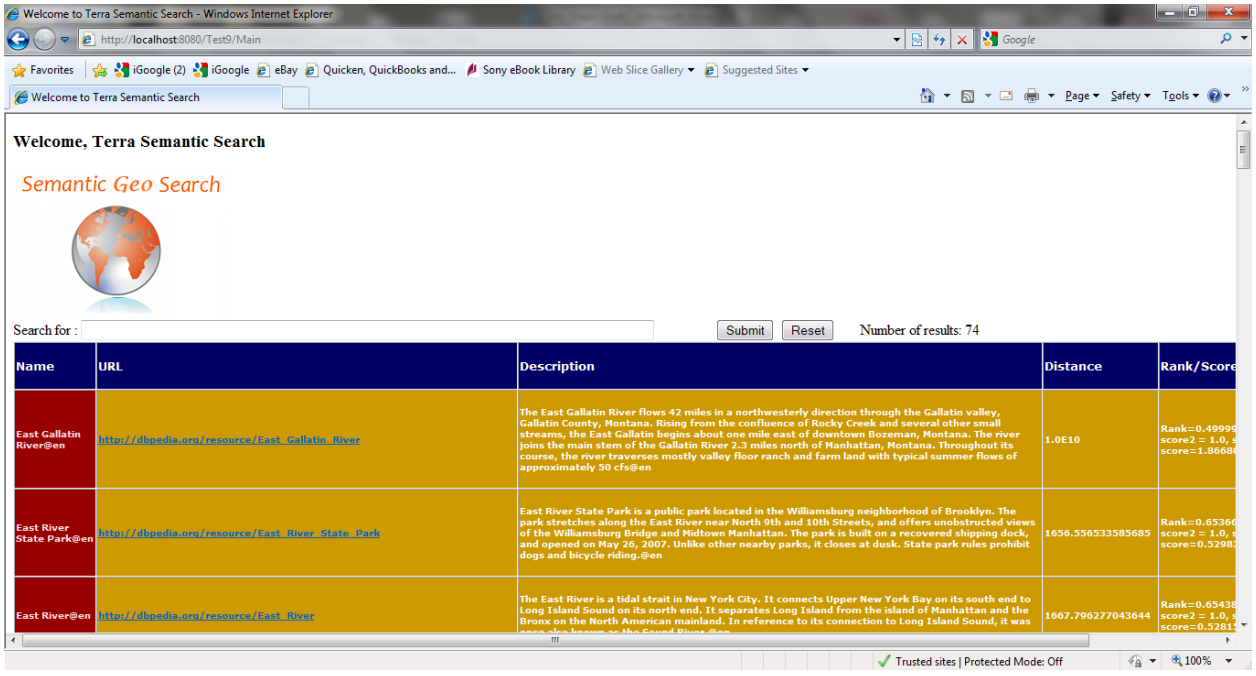

Figure 7-5: SGSS user interface showing the results with their ranking

Chapter 8. Evaluation

8.1 Methodology:

The semantic geo search service (SGSS) evaluation was conducted in comparison with a keyword-based search engine. For this purpose a new search service was developed that uses a keyword-based search and the known ranking algorithm TF-IDF (Term Frequency and Inverse Document Frequency). This keyword based algorithm is widely used in Web crawling and ranking of Web pages. Moreover, it is used in few semantic search engines like Falcons [CGQ08] in parallel with other factors to rank the Web search results. A group of users was asked to conduct the evaluation.

There are several reasons why we developed a search service and did not use an existing search service from the Web:

1- Most of the existing Web search engines are based on crawling Web pages, retrieving the relevant documents and ranking them. However, our SGSS is not a Web crawler but is intended for existing, semantic, well structured knowledge base. So, comparing two services with different data sources might give a better score for the service that is using the richer data source. This is because rich data sources will find many relevant results, not because the algorithm used in searching is more powerful, but because of the nature of the existing data. Therefore, we decided to develop the keyword based search service and have it using the same data source used by SGSS.

2- While the SGSS cares more about the geo-information and presenting it to the user, Web crawlers do not care about this feature and will always provide HTML pages with relevant information. Comparing a geo-search service with a Web crawler is not fair as the crawler will be assigned a lower score by users looking for geo-information which does not always exist in HTML pages.
3- Since most of the existing keyword based search engines are supported by commercial companies like *YAHOO and MSN* for the purpose of profit, they have deployed more powerful and advanced processing hardware than we did for our educational and research search engine. Consequently, comparing a super computer crawling billions of pages, with a workstation running on a local Web host is not fair especially in terms of processing time.

The traditional search service was developed and integrated with TF-IDF ranking program to produce the final ranked results. Since the relation phrases in such a keyword algorithm are not necessary in fact it produces irrelevant results, we filter out relation phrases from the user query and keep only the keywords like places and locations.

8.2 Developing A Keyword-Based TF-IDF Search Service:

The TF-IDF is used to compute two scores for each result. Calculating the importance of a returned result is based on how many times the query keywords have appeared in the returned text of the result. The importance of the result is proportional to the number of times the keywords appeared in the result but is controlled by IDF. IDF reduces the importance of the

score if a given keyword appeared so many times in the whole set of results thereby reducing the effect of such very common words.

Consider if we have a set of documents D = $\{d_1, d_2, d_3 .. d_n\}$ where each document represents a single result returned by the search engine and consider that the user query consists of the following set of terms T = $\{t_1, t_2, t_3 \ldots t_m\}$. Each TF-IDF value represents a score for a single document for a single term.

The TF [SM86] value for a result document d_i with regard to a term t_j is given by:

$$TF_{i,j} = \frac{t_{j\ i}}{|d_i|}$$

Where $t_{j\ i}$ is the number of times the term t_j was found in the result document d_i and $|d_i|$ is the number of terms in d_i.

As mentioned earlier, IDF [SM86] value intends to balance the appearance of the term in a document with regard to its overall importance. That overall importance is measured based on the number of documents in which this term has ever appeared.

$$IDF = \log \frac{|D|}{1 + |TF_{nonZero}|}$$

Where $TF_{nonZero}$ is the number of times the TF value was calculated and found greater than zero which equals to the number of documents in which the term t_j was found at least one time.

So, the more a term t_j appears in D, the less effect it is given in judging the importance of any document in D. The TF and IDF scores are combined to represent the overall TF-IDF [SM86]

score, which defines the total score value that is used in calculating the rank value. The TF-IDF value for a document d_i is given by:

$$TFIDF_{i,j} = TF_{i,j} \times IDF$$

As we notice from the above equation that a high TF-IDF value is obtained from a big frequency number of a term in a single document and a low TF-IDF value is obtained from a big frequency number of a term in the whole set of documents D.

Since the above equations for TF and IDF are calculated for a single document for a single term, it is necessary to show how the overall calculation method is being used and applied to all the terms together to calculate the total score of the document d_i. The total score for a document d_i with regard to the set of query terms in T can be calculated as:

$$TotalScore_i = \sum_{k=1}^{k=m} TFIDF_{i,k}$$

Now, the raking value for d_i can be computed by:

$$Rank(d_i) = \frac{1}{totalScore_i}$$

So, the bigger the total score is, the better rank is given to d_i.

Figure 8-1 explains briefly the process of ranking a set of documents according to TF-IDF architecture.

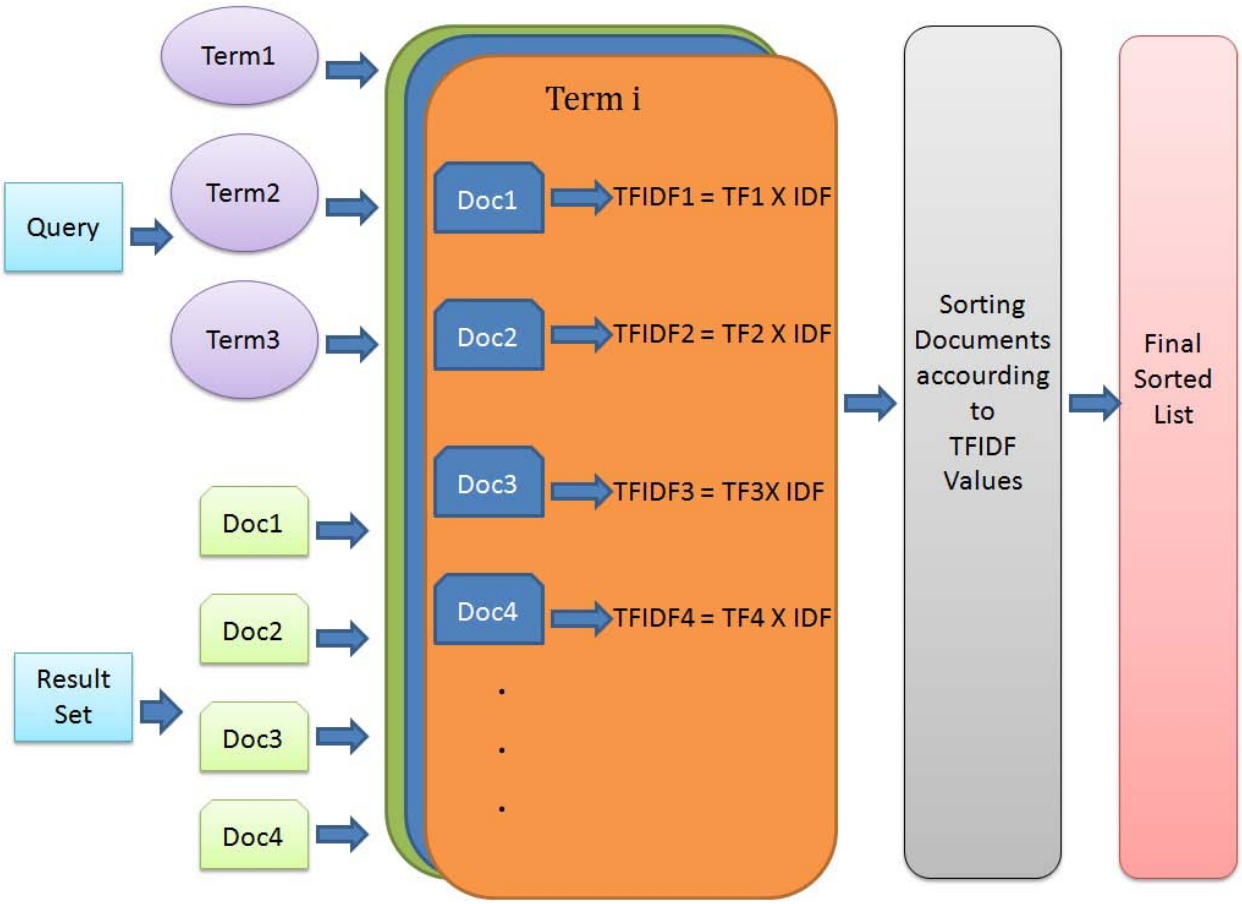

Figure 8-1: General architecture of TF-IDF ranking.

TF-IDF algorithm:

Procedure TFIDF_RANK

Inputs:

- A set of result documents D.
- A set T of query keywords.

Output:

The list of ranked documents D.

Begin.

/* to store the total number of docs in which the term appeared. */

Define variable *totalTermAppearance*

/* to store a temporary value of the document term frequency */

Define variable *TF*

/* to store the value of IDF */

Define variable *IDF*

Loop through T.

Current T element → t_j

totalTermAppearance =0

IDF =0

Loop through D.

Current D element → d_i

TF = $CALC\_TF(d_i, t_j)$.

If (*TF*>0) Then, *totalTermAppearance* ++

$d_i.tempTF$ = TF

End loop.

/* calculate IDF for this term and use it to calculate TFIDF */

IDF = $\log \frac{|D|}{1+totalTermAppearance}$

Loop through *D*.

Current D element → d_i

$$d_i.totalTFIDF = d_i.totalTFIDF + (IDF \times d_i.tempTF)$$

End loop.

End loop.

Loop through *D*.

Current D element → d_i

$d_i.rank$ = 1/($d_i.totalTFIDF$)

End loop.

MergeSort(*D*) /*sort D according to the rank value */

Return *D*.

End TFIDF_RANK.

Procedure CALC_TF(d, t).

/*to store the frequency value of the term */

Define variable *frequency*=0

All words in d ➔ *W*

loop through *W*

Current word → w_i

If w_i == t, Then *frequency*++.

End loop.

Return frequency/|W|.

End CALC_TF.

The above algorithm costs $(|T| \times N^2 \times |totalWords|) + N + N\log N$, where N = |D| and *totoalWords* is the sum of all words in all documents of the set D.

➔ Total cost is O($|T| \times N^2 \times |totalWords|$)

Below is the user interface of the developed search service. It is very similar to the SGSS interface, where the user can enter a query and receives a list of ranked results according to the

TF-IDF algorithm. The rank, URL, name and description are all part of the informative data describing a single result.

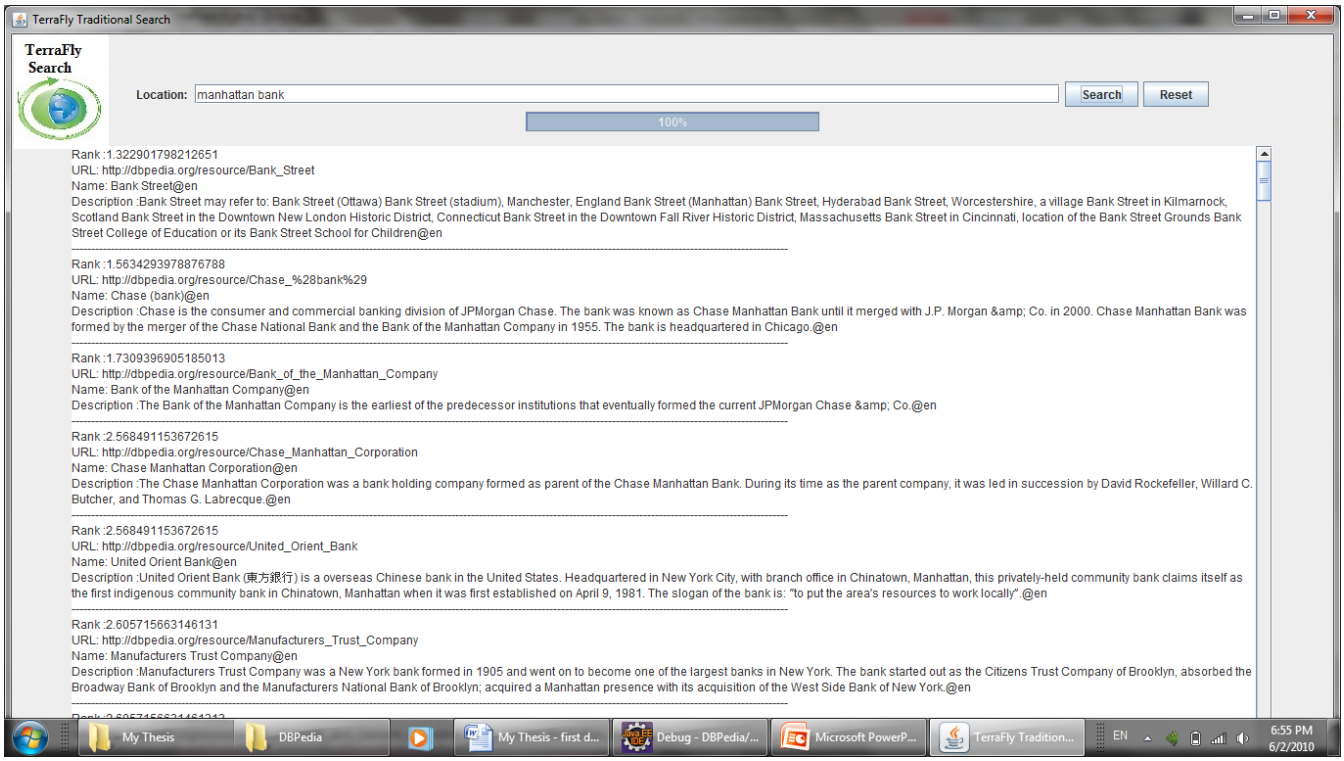

Figure 8-2: User interface of the developed keyword TF-IDF based search engine.

8.3 User Evaluation:

The evaluation was based on user input. Ten users were given a list of evaluation factors as listed below, in order to assign a score from 1 to 10 for each factor for each query test. The scores for each factor for each user are listed and summed as shown below for both search services.

Table 8-1: Traditional Keyword-Based Search Scores assigned by testing users.

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10
Accuracy	65	90	40	92	90	40	7	62	20	80
No. of relevant results	60	10	33	36	61	30	70	30	10	70
No. of retrieved results	174	55	71	54	140	70	102	90	65	109
Unpredictable but useful results	80	10	10	10	30	9	10	50	10	9
General Evaluation	65	20	80	99	80	12	90	40	70	70

Total retrieved documents =930

Total relevant documents = relevant results + unpredictable but useful results = 410+228=638

Table 8-2: SGSS Scores assigned by testing users.

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10
Accuracy	88	94	48	77	100	43	10	90	40	70
No. of relevant results	60	10	50	70	70	46	90	20	9	50
No. of retrieved results	136	56	90	96	97	60	131	80	44	96
Unpredictable but useful results	70	10	30	21	20	9	30	50	20	20
General	73	50	87	99	92	20	90	32	60	90

Evaluation										

Total retrieved documents =886

Total relevant documents = relevant results + unpredictable but useful results = 475+280= 755

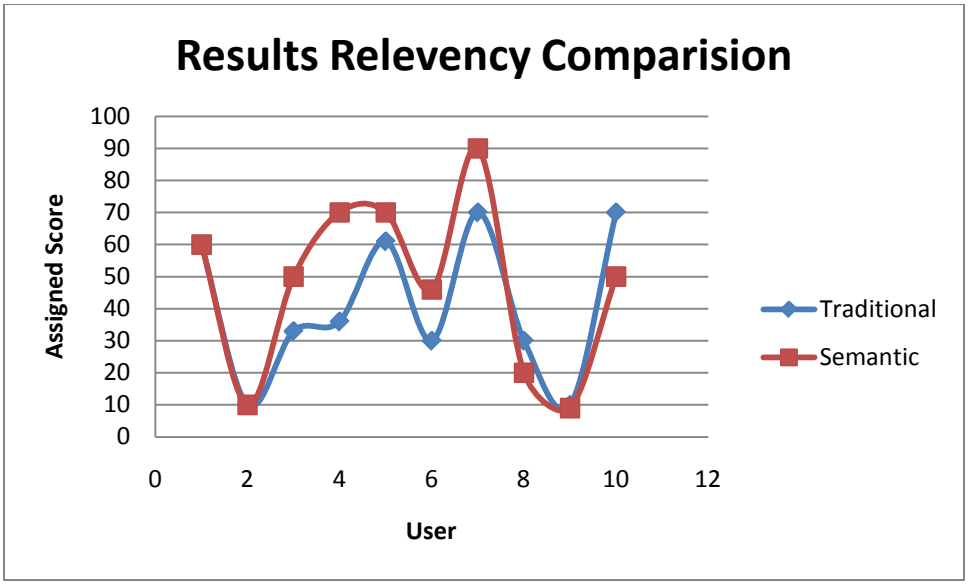

Figure 8-3: Results relevancy comparison.

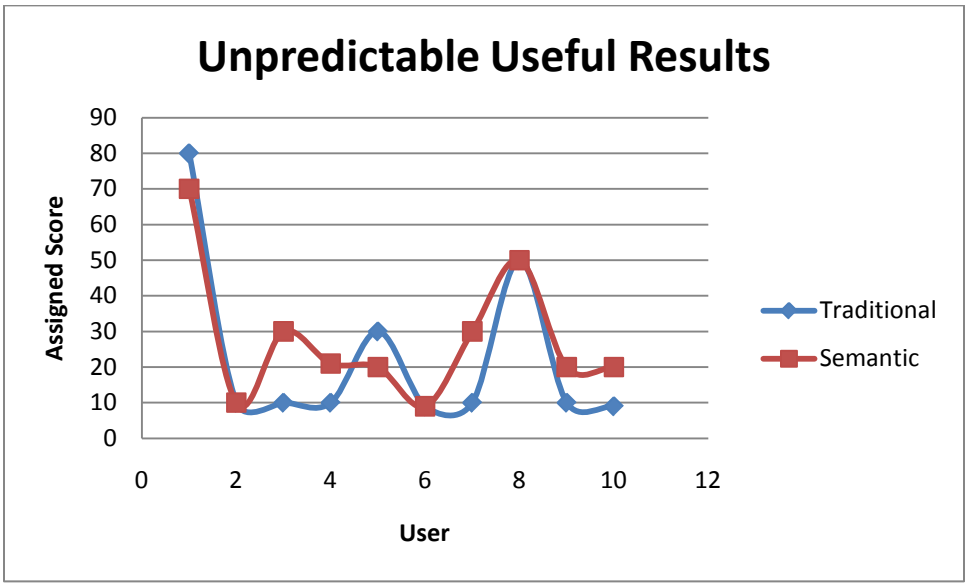

Figure 8-4: Unpredictable but useful results comparison.

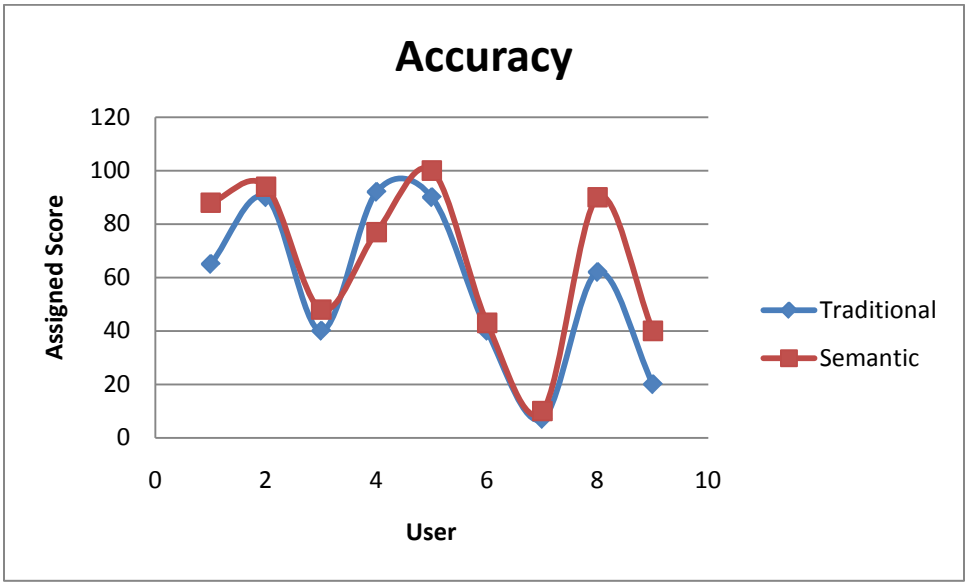

Figure 8-5: Accuracy comparison.

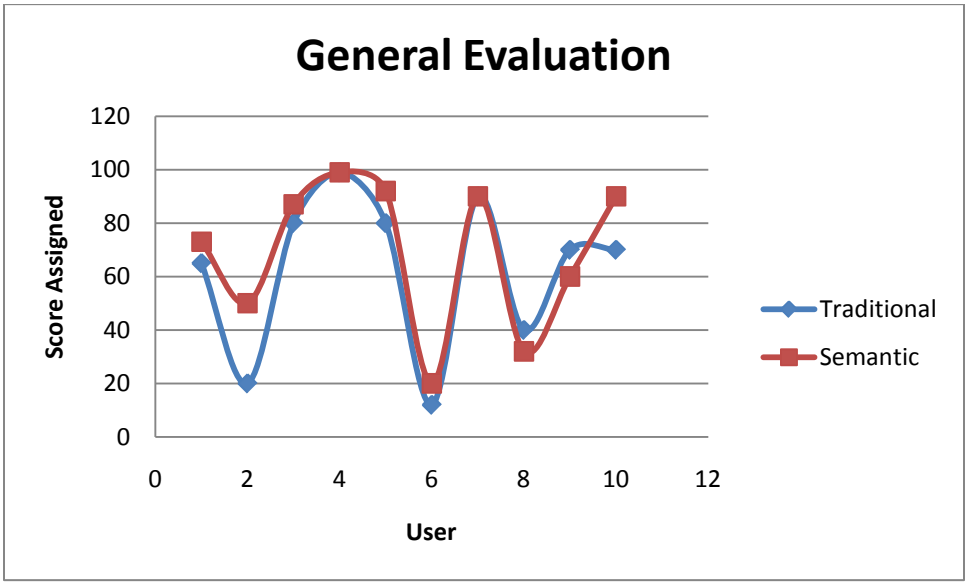

Figure 8-6: General user evaluation of SGSS versus the traditional search.

8.4 Precision:

Precision is defined as the proportion of the total number of related documents retrieved to the number of documents retrieved [ZZGZ08]. The highest precision value is 1 which reflects that every result retrieved by the search is relevant. However, it does not indicate the magnitude of the retrieved results from the available related documents.

$$Precision = \frac{No.of\ retrieved\ related\ documents}{No.of\ retrieved\ documents}$$

From the above user evaluation we conclude that the traditional keyword search receives a precision of:

$$\frac{638}{930} = 0.6$$

On the other hand, the semantic geospatial search service receives a precision of:

$$\frac{755}{886} = 0.85$$

The semantic search service has proved its higher accuracy than the traditional keyword-based search. Our interpretation for this can be summarized as follows:

- SGSS applies a mechanism to identify the geographical entities and give them a higher priority than other documents only meeting the query terms. Therefore, retrieving geographical resources improves the accuracy over retrieving everything.
- SGSS considers the geospatial distance in both searching the knowledge base and ranking. Therefore, calculating the distance between the user query location and resource location helps in ranking the closest resources. For many users, retrieving places which are thousands of miles away from his/her location is an accuracy issue.

The accuracy could even improve if we could calculate the distances for all the resources but this is not possible because the coordinate information is not always available in the knowledge base.

- The semantic search considers the internal structure of the target concept in both searching and ranking. Resources with more properties represented by their concept receive a higher internal structure score. This leads to a higher accuracy than considering only the keywords which might appear in irrelevant document intensively discussing the query keywords.

8.5 Recall:

Recall is defined as the proportion of the total number of retrieved related documents to the number of all related documents in the system [ZZGZ08]. The highest recall value is 1 which tells that every related document in the system was retrieved. However, no information here about how many non relevant results was also retrieved by the search.

$$Recall = \frac{No.of\ retrieved\ related\ documents}{No.of\ related\ documents\ in\ system}$$

We are not able to exactly calculate the number of the related documents in the system for each query and we doubt anybody is able to do so. However, if we assume that the total number of documents for all the tested queries in the system equals to a number N. Then, it is clear that the recall value for the keyword-based search equals to $\frac{638}{N}$ compared to the one for SGSS which equals to $\frac{755}{N}$. It is obvious that SGSS will produce a higher recall than the traditional keyword-based search. Our interpretation can be summarized as follows:

As the semantic search enriches the query by adding super classes, subclasses, equivalencies, synonyms, hyponyms, hypernyms and the inferred classes, the query is enriched with more terms. As a result, relevant documents that have not been stated by the user in the query are being extracted and discovered using the ontological processing and reasoning. This process enriches the retrieval task with more results and adds more hidden resources which contribute to the overall recall of the search engine. Moreover, since SGSS searches for places around the site specified by the user, it discovers the desired places using geo coordinates even if those places are not annotated with a geo location. On the other hand, the traditional keyword-based approach searches for resources with exact location terms.

The evaluation can produce different results when we assign different scoring weights for the different scores. However, perfect scoring weights need a larger group of users and more experiments.

8.6 Response Time:

The response time does not accurately evaluate the performance of our SGSS or the traditional search algorithm because of the external factors involved like the response time of the external knowledge base like DBPedia. DBPedia server is fluctuating in responding to our queries regardless of the query structure or complication. Different response times might be obtained for the same query when it is run on DBPedia at different times. However, users have found the traditional search algorithm faster than SGSS by a factor of 4. Our interpretation for this performance issue is because of the extra operations SGSS executes like Geo-query

preparation, gazetteer lookup, ontology processing, reasoning and the complexity of the semantic query unions

Chapter 9. Summary

A general and extendible methodology was presented for formalizing a user query to search for geospatial information in semantic data sources. The formalization process included parsing, meaning retrieval, ontological reasoning on meanings to retrieve implicit classes, formalizing the SPARQL query, and ranking the results.

An infrastructure was developed that can be adopted and extended by any semantic geo-search engine. It included integration among different components like Protégé OWL, Jena API, external SPARQL endpoints, Pellet reasoner and different data sources and gazetteers.

DBPedia high level ontology was integrated with another one created from Geonames codes in order to enrich the geospatial knowledge systematically. The ontology was also used to measure the internal structure of a retrieved resource and how well it meets the structure defined in the ontology. Semantic matching of the terms along with the internal structure was measured as scoring criteria in the ranking engine.

The inconsistencies arising from converting a text-based knowledge into RDF-based one using information extractors motivated the creation of an approximation mechanism. It was used to identify geo-resources that are not annotated correctly, or that are missing important geo-annotations.

The ranking engine considered mainly three scores: the matching of the resource terms with the query semantic terms, the spatial distance between the resource location and the query location, and the internal structure match.

Finally, for evaluation a keyword-based search service was developed based on TF-IDF ranking. The search service was compared with SGSS and several evaluation criteria were considered: relevant documents, accuracy, unpredictable but useful results, response time and user general opinion. The results showed that the SGSS outperformed the keyword-based search for both accuracy and recall. However, several factors impacted the overall performance and evaluation of SGSS, such as the consistency of the underlying knowledge base and the ranking scoring weights that could improve the accuracy significantly. However, the keyword-based search was found faster than SGSS due to the complexity of the SGSS execution steps.

Chapter 10. Future Work

The general query structure *(Place-Relation-Site)* has been proposed to help the user define exactly what he/she needs and to help the system define the requirements of the user precisely. However, a simple query parser was used and in future work a more sophisticated one we will applied that can extract composed relations and locations. For example, "*Hotels close to airport in San jose*" has two relations "*close to*" and "*in*" which needs a parser and a reasoner to interpret the geo-area based on dividing the assumptions into rules and to reason them out. Also, the parser can be improved to identify the location phrase that contains description of the location not only the name.

Two ontologies were integrated into the DBPedia basic ontology namely the Directory and the geo-relations ontologies. The separation between geo relations and concepts needs an improvement whereby a domain expert can integrate both ontologies for a better inference mechanism. Inferences about both relations and concepts from one ontology provide a complete picture about whether required data exist in the system as illustrated next.

The nature of our system implied that the T-Box and A-Box, were used having box stored separately in two different servers. On the one hand, a general ontology (DBPedia) was used that integrated more concepts and classes to create an overall richer geo-ontology and it was stored in a local system. On the other hand, the DBPedia server was the main A-Box source thereby making it hard to satisfy some reasoning requirements where T-Box and A-Box were

required to be part of a one reasoning cycle. For example, rather than defining the potential locations from the query "*Doctors south of Miami*" , it should be possible to run a SPARQL query to retrieve all the doctor offices stored in the ontology where these locations satisfy the predicate *(?uri Geo:southOf "Miami"*).

Although DBPedia is a linking hub that connects many data sources and many of them are geospatial like Geonames, DBPedia does not record each and every single geo-entity in the world. This is because the main source of data is Wikipedia which in itself is not a specialized geo-database but it does contain and link a huge amount of geo information. Missing geo data from DBPedia includes for example "*PizzaHut*" branches in each and every street of a town. Moreover, small businesses and shops are very rarely to be found in DBPedia. In future work there need to be more sources of geo-data to be integrated with the system.

Ranking was applied as a set of scores each contributing to the overall ranking value. This approach seems fair in most of situations, since ranking can best be evaluated by users as it is strongly connected with the user's desires. Many search engines do not provide preferences for users to play a role in the ranking criteria. Relying on the system to completely define the best rank can be enhanced by involving users to assign weights to the ranking criteria. For example, a female user might want any "*beauty shop*" or a similar facility no matter what the distance is. Such a user can assign a heavier weight to the score of the place semantic terms than to the score of spatial distance. Therefore, our future work will improve the interface for ranking purposes.

Finally, due to performance issues only part of the information was retrieved that was related to describing the geo-entity resulting from user query. The semantic search engine should be different than the traditional one in presenting results in such a way that it can link the user to other related ideas and objects on the Web. The "*owl:sameAs*" and "*owl:seeAlso*" can help in achieving this which will be yet another point of improvement

References

[ABK+08] Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a Web of open data. *In Proceedings of the 6th International SemanticWeb Conference (ISWC),* volume 4825 of Lecture Notes in Computer Science, pages 722-735. Springer, 2008.

[ADL+09] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: light-weight linked data publication from relational databases. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 621-630. ACM, 2009.

[AKT08] A. Alam, L. Khan, and B. Thuraisingham. Geospatial resource description framework (grdf) and security constructs. *In Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference* on, pages 475-481, 7-12 2008.

[ALH09] Soren Auer, Jens Lehmann, and Sebastian Hellmann. Linkedgeodata: Adding a spatial dimension to the Web of data. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009, volume 5823 of Lecture Notes in Computer Science*, pages 731-746. Springer,2009.

[AMS05] Kemafor Anyanwu, Angela Maduko, and Amit Sheth. Semrank: ranking complex relationship search results on the semantic Web. In WWW '05: *Proceedings of the 14th international conference on World Wide Web*, pages 117-127, New York, NY, USA, 2005. ACM.

[ASJ07] A. I. Abdelmoty, P. Smart, and C. B. Jones. Building place ontologies for the semantic Web:: issues and approaches. In GIR '07: *Proceedings of the 4th ACM workshop on Geographical information retrieval*, pages 7-12, New York, NY, USA, 2007. ACM.

[AvH08] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. MIT Press, Cambridge, MA, 2. edition, 2008.

[BB09] Christian Becker and Christian Bizer. Exploring the geospatial semantic Web with dbpedia mobile. *J. Web Sem.*, 7(4):278-286, 2009.

[BCC+08] Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. Hybrid search: Effectively combining keywords and ontology-

based searches. In Manfred Hauswirth, Manolis Koubarakis, and Sean Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg, June 2008. Springer Verlag.

[BCCW05] Albert Bifet, Carlos Castillo, Paul A. Chirita, and Ingmar Weber. An analysis of factors used in a search engine's ranking. In *First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[BCT07] Karin Breitman, Marco Antonio Casanova, and Walt Truszkowski. *Semantic Web Concepts, Technologies and Applications*. NASA Monographs in Systems and Software Engineering. Springer-Verlag New York, Inc., 2007.

[BHBL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1-22, 2009.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic Web. *Scientific American*, 284(5):34-43, 2001.

[CGQ08] Gong Cheng, Weiyi Ge, and Yuzhong Qu. Falcons: searching and browsing entities on the semantic Web. In WWW '08: *Proceeding of the 17th international conference on World Wide Web*, pages 1101-1102, New York, NY, USA, 2008. ACM.

[DBP10] DBPedia. The DBpedia Knowledge Base, 2010. url: http://dbpedia.org/About.

[Ege02a] Max J. Egenhofer. Toward the semantic geospatial Web. In Agns Voisard and Shu-Ching Chen, editors, *ACM-GIS*, pages 1-4. ACM, 2002.

[Ege02b] Max J. Egenhofer. Toward the semantic geospatial Web. In *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 1-4, New York, NY, USA, 2002. ACM.

[FEAC02] F. Fonseca, M. Egenhofer, P. Agouris, and C. Camara. Using Ontologies for Integrated Geographic Information Systems. *Transactions in GIS*, 6(3), 2002.

[FIU10] FIU. Semantic SQL, 2010. url:http://semanticsql.com.

[GDG10] Carlos Granell, Laura Daz, and Michael Gould. Service-oriented applications for environmental models: *Reusable geospatial services. Environmental Modelling and Software*, 25(2):182-198, 2010.

[Geo10] Geonames. Geonames, 2010. url:*http://www.geonames.org/*.

[GHM+08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. Owl 2: The next step for owl. *Web Semant.*, 6(4):309-322, 2008.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199-220, 1993.

[Guh09] R. Guha. Toward the intelligent Web systems. *In Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, pages 459-463, 23-25 2009.

[Hav03a] Taher H. Haveliwala. Topic-sensitive pagerank: A context sensitive ranking algorithm for Web search. IEEE *Transactions on Knowledge and Data Engineering*, 15:784-796, 2003.

[Hav03b] T.H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for Web search. *IEEE Transactions on, Knowledge and Data Engineering.* 15(4):784 - 796, july-aug. 2003.

[HLA09] Sebastian Hellmann, Jens Lehmann, and Sren Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal on Semantic Web and Information Systems*, 5(2):25-48,2009. To be published.

[Hor05] Ian Horrocks. Owl rules, ok? *In Rule Languages for Interoperability*. W3C, 2005.

[HPDC09] LP Hewlett-Packard Development Company. Jena A Semantic Web Framework for Java, 2009. url:*http://jena.sourceforge.net.*

[HR04] K. Hiramatsu and F. Reitsma. Georeferencing the semantic Web:Ontology-based markup of geographically referenced information. *In Proc. of the Joint EuroSDR /EuroGeographics Workshop on Ontologies and Schema Translation Services*, 2004.

[Jar89] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. Journal of the American Statistical Association 84:414-420, 1989.

[JZR+08] Rosie Jones, Wei Vivian Zhang, Benjamin Rey, Pradhuman Jhala, and Eugene Stipp. Geographic intention and modification in Web search. *International Journal of Geographical Information Science*, 22(3):229-246, 2008.

[KBM08] Vipul Kashyap, Christoph Bussler, and Matthew Moran. *The Semantic Web*: *Semantics for Data and Services on the Web*. Springer, Berlin, 2008.

[KFNM04] Holger Knublauch, Ray W. Fergerson, Natalya Fridman Noy, and Mark A. Musen. The protege owl plugin: An open development environment for semantic Web applications. In *International Semantic Web Conference*, pages 229-243, 2004.

[KOD09] M. Kassab, O. Ormandjieva, and M. Daneva. An ontology based approach to non-functional requirements conceptualization. *In Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on*, pages 299 -308, sept. 2009.

[KRSW08] Gjergji Kasneci, Maya Ramanath, Fabian Suchanek, and Gerhard Weikum. The yago-naga approach to knowledge discovery. *SIGMOD* Rec., 37(4):41-47, 2008.

[LA04] R. Lemmens and H. Arenas. Semantic matchmaking in geo service chains: reasoning with a location ontology. In Database and Expert Systems Applications, 2004. *Proceedings. 15th International Workshop* on, pages 797 - 802, aug.-3 sept. 2004.

[LH08] Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the alc description logic. In *ILP'07: Proceedings of the 17th international conference on Inductive logic programming*, pages 147-160, Berlin, Heidelberg, 2008. Springer-Verlag.

[LK08] Jens Lehmann and Sebastian Knappe. Dbpedia navigator. Semantic Web Challenge, International Semantic Web Conference 2008, 2008.

[LLC06] SoftCorporation LLC. Suggester Spellcheck Spell Checking Java library, 2006. url:*http://www.softcorporation.com/products/spellcheck/.*

[LSD09] Fabrizio Lamberti, Andrea Sanna, and Claudio Demartini. A relation-based page rank algorithm for semantic Web search engines. *IEEE Trans. on Knowl. and Data Eng.*, 21(1):123-136, 2009.

[LUM06] Yuangui Lei, Victoria Uren, and Enrico Motta. Semsearch: A search engine for the semantic web. In Proc. 5th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks, Lect. Notes in Comp. Sci., Springer, Podebrady, Czech Republic, pages 238-245. Springer-Verlag, 2006.

[LWH07] Yufei Li, Yuan Wang, and Xiaotao Huang. A relation-based search engine in semantic Web. *IEEE Trans. on Knowl. and Data Eng.*, 19(2):273-282, 2007.

[LY08] Wenwen Li and Chaowei Yang. A semantic search engine for spatial Web portals. *In Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International,* volume 2, pages II-1278 –II-1281, 7-11 2008.

[MG10] MindSwap Group. Mindswap Group, 2010. url: http://www.mindswap.org.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[Pro03] Protg Project. The Protg Ontology Editor and Knowledge Acquisition System, 2003. urlhttp://protege.stanford.edu/.

[PS08] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C recommendation, W3C, January 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[RCC+03] Naphtali Rishe, Maxim Chekmasov, Marina Chekmasova, Scott Graham, and Ian De Felipe. On-demand geo-referenced TerraFly data miner. In IUI '03: *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 277-279, New York, NY, USA, 2003. ACM.

[REW05] REWERSE Project. Reasoning on the Web with Rules and Semantics, 2005. *url:http://rewerse.net/publications/rewersepublications.html#REWERSE-DEL-2005-A1-D2.*

[RK08a] S.K. Rajapaksha and N. Kodagoda. Internal structure and semantic Web link structure based ontology ranking. In *Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference* on, pages 86 - 90, dec. 2008.

[RK08b] U.U.S.K. Rajapaksha and N. Kodagodat. Semantic Web search and ontology ranking algorithm. *SEMANTIC WEB SEARCH PSLIIT*, 2:25-29, 2008.

[RSC+04] N. Rishe, Y. Sun, M. Chekmasov, A. Selivonenko, and S. Graham. System architecture for 3d TerraFly online gis. pages 273 - 276, dec. 2004.

[SAMA+05] A. Sheth, B. Aleman-Meza, F. S. Arpinar, Amit Sheth, C. Ramakrishnan, C. Bertram, Y. Warke, K. Anyanwu, Boanerges Aleman-meza, I. Budak Arpinar, , K. Kochut, C. Halaschek, Cartic Ramakrishnan, Yashodhan Warke, D. Avant, F. Sena Arpinar, Kemafor Anyanwu, and Krys Kochut. Semantic association identification and knowledge discovery for national security applications. *Journal of Database Management*, 16:33-53, 2005.

[SBG+06] M. Sabou, C Baldassarre, L Gridinoc, S Angeletou, Enrico Motta, M. d'Aquin, and M. Dzbor. Watson: A gateway for the semantic Web. In *ESWC 2007 poster session*, June 2007-06.

[SEM10] SemanticFocus. Semantic Web Layer Cake, 2010. Url: http://www.semanticfocus.com/media/insets/semanticWeb-layer-cake.png.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

[ST07] Arno Scharl and Klaus Tochtermann. *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[TDG07] P. Top, F. Dowla, and J. Gansemer. A dynamic programming algorithm for name matching. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 547-551, 1 2007-april 5 2007.

[TK02] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res*., 2:45-66, 2002.

[TSB09] D. Tumer, M.A. Shah, and Y. Bitirim. An empirical evaluation on semantic search performance of keyword-based and semantic search engines: Google, yahoo, msn and hakia. In *Internet Monitoring and Protection, 2009. ICIMP '09. Fourth International Conference* on, pages 51-55, may 2009.

[Uni10a] FIU University. TerraFly, 2010. url:http://www.terrafly.com.

[Uni10b] Princeton University. WordNet, 2010. url:http://wordnet.princeton.edu/wordnet/.

[VBGK09] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the Web of data. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *International Semantic Web Conference, volume 5823 of Lecture Notes in Computer Science*, pages 650-665. Springer, 2009.

[Wel05] James Welton. *A manual of logic*. University Tutorial Press, second edition, 1905.

[Wik10] Wikipedia. DBpedia, 2010. url:*http://en.wikipedia.org/wiki/DBpedia*.

[W3C04] W3C. OWL Web Ontology Language Reference, 2004. url: *http://www.w3.org/TR/owl-ref/*.

[Win99] William E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

[XZJL08] Songhua Xu, Yi Zhu, Hao Jiang, and Francis C. M. Lau. A user-oriented Webpage ranking algorithm based on user attention time. In Dieter Fox and Carla P. Gomes, editors, AAAI, pages 1255-1260. *AAAI Press*, 2008.

[YRL09] Xing Yi, Hema Raghavan, and Chris Leggetter. Discovering users' specific geo intention in Web search. In WWW '09: *Proceedings of the 18th international*

conference on World wide Web, pages 481-490, New York, NY, USA, 2009. ACM.

[ZLT+08] Qian Zhong, Juanzi Li, Jie Tang, Yi Li, and Lizhu Zhou. Path similarity based directory ontology matching. In *Web-Age Information Management, 2008. WAIM '08. The Ninth International Conference* on, pages 37-44, 20-22 2008.

[ZZGZ08] Guobing Zou, Bofeng Zhang, Yanglan Gan, and Jianwen Zhang. An ontology-based methodology for semantic expansion search. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on,* volume 5, pages 453-457, oct. 2008.