OntoClass : A TOOL FOR ONTOLOGY CATEGORIZATION

by

ARPAN SHARMA

(Under the direction of Krzysztof J. Kochut)

ABSTRACT

The Semantic Web envisions making World Wide Web content machine processable, not just readable by human beings. Ontologies form the backbone of the Semantic Web. As ontologies are increasingly coming into existence, their evaluation is gaining importance. Ontology Alignment is the process of determining correspondences between concepts of ontologies describing the same domain or similar domains.

For effective alignment of ontologies, computation of equivalent elements is not sufficient. By finding the degree or measure of similarity of a given target ontology with a large reference ontology such as an ontology constructed out of Wikipedia, the topical content of the target ontology can be categorized into a particular domain. Ontology categorization is a way of evaluating a given ontology by indicating how well a given ontology describes the domain it belongs to. The proposed approach attempts to compare a given target ontology and determine how it fits within the ontology constructed out of Wikipedia and categorizes the target ontology into a particular domain of knowledge.

INDEX WORDS : Ontology, Ontology Categorization, Ontology Alignment, Ontology Subsumption, Ontology Matching.

OntoClass : A TOOL FOR ONTOLOGY CATEGORIZATION

by

ARPAN SHARMA

B.E., University of Mumbai, India, 2001

M.C.M, University of Pune, India, 2004

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2010

OntoClass : A TOOL FOR ONTOLOGY CATEGORIZATION

by

ARPAN SHARMA

Major Professor: Krys Kochut

Committee: John Miller
Budak Arpinar

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2010

DEDICATION

Dedicated to my major professor Dr. Krzysztof J. Kochut who has always encouraged me throughout my coursework and research, my family - my grandmother, parents and brother who have always been my greatest inspiration and my husband whose unconditional love and support made this work possible.

ACKNOWLEDGEMENTS

I am extremely grateful to my advisor Dr. Krzysztof J. Kochut for his guidance and support throughout my research and academic study at UGA. I would like to thank Dr. John A. Miller and Dr. Budak Arpinar for serving on my advisory committee.

A special thanks to Ms. Chandana Mitra for her continuous encouragement, love and support in Athens. I am highly grateful to my friends Ujwal, Priyanka and Sanjeev for all their love and support. I owe special thanks to Swapnil, my husband, who has always encouraged me.

I would also like to thank my past and current lab-mates Maciej Janik, Matthew Eavenson and Sonu Swaika for their help throughout my research at UGA.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 SEMANTIC WEB

The rapid development of the Web technology has brought along an increasing interest in research on knowledge sharing in a distributed environment. Semantic Web is an evolving development of the World Wide Web in which the meaning (semantics) of information and services on the Web is defined, making it possible for the Web to "understand" and satisfy the requests of people and machines to use the Web content. Humans are capable of performing various tasks such as booking a flight, searching a product with the lowest price, finding a similar word for a given word, etc. However, computers cannot perform these tasks without human intervention because they cannot really understand and interpret the meaning of the words sent on the Web. This is where the Semantic Web steps in. The Semantic Web is the future generation in WWW technology and the transition from Web 2.0 to Web 3.0.

Tim Berners-Lee originally expressed the vision of the Semantic Web as follows:

“I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.” [1].

There is a lot of data on the Web that we use every day, but is not a part of the Web. Data are controlled by the applications on the Web and hence it is limited to those applications. The

Semantic Web extends principles of Web from documents to data. Data should be accessed with a URI, but should be related to each other as documents. The Semantic Web provides a framework for sharing and reusing data across various application boundaries and provides tools to access the data manually as well as automatically. This helps in discovering new relationships between pieces of data. Hence, the Semantic Web includes a Web of data.

Web pages are usually created using Hypertext Markup Language (HTML) which is a markup language to embed different objects in a webpage. The meta and link elements in HTML can be used to add metadata to an HTML page. In Semantic Web terms, this is equivalent to the process of defining RDF relationships for that page as a "source". Note, however, that these elements can be used to define relationships for the enclosing HTML file only, whereas the Semantic Web allows the definition of relationships on any resource on the Web. With HTML, there is no way to prove that particular pieces of data are bound to each other with certain relationships and are distinct from other objects on the page.

HTML describes documents and the links between them. Data that is generally hidden away in HTML files is often useful in some contexts, but not in others. The problem with the majority of data on the Web that is in this form at the moment is that it is difficult to use on a large scale, because there is no global system for publishing data in such a way as it can be easily processed by anyone. For example, information about local sports events, weather information, plane times, television guides, all of this information is presented by numerous sites, but all in HTML. The problem with that is that, is some contexts, it is difficult to use this data in the ways that one might want to do so. So the Semantic Web can be seen as a huge engineering solution. A large number of Semantic Web applications can be used for a variety of different tasks, increasing the modularity of applications on the Web [2].

The Semantic Web is generally built on syntaxes which use URIs to represent data, usually in triples based structures: i.e. many triples of URI data that can be held in databases, or interchanged on the World Wide Web using a set of particular syntaxes developed especially for the task. Semantic Web involves publishing in languages specifically designed for data: Resource Description Framework (RDF), Web Ontology Language (OWL), and Extensible Markup Language (XML). RDF, OWL, and XML, by contrast, can describe arbitrary things such as people, meetings, or airplane parts. An example of a tag that would be used in a non-semantic web page [2]:

<place>Georgia</place>

Encoding similar information in a Semantic Web page might look like this:

<place rdf:about="http://dbpedia.org/resource/Georgia">Georgia</place>

The RDF statement above describes a resource with URI http://dbpedia.org/resource/Georgia having a property called “place” and corresponding property value “Georgia”.

The Semantic Web is an extension of the current Web and not its replacement. Islands of RDF and possibly related ontologies can be developed incrementally. Major application areas (like Health Care and Life Sciences) may choose to “locally” adopt Semantic Web technologies, and this can then spread over the Web in general. The Semantic Web envisions making Web content machine processable, not just readable or consumable by human beings. It aims at creating a world of software agents that understand documents semantically in a decentralized architecture. This is accomplished by the use of ontologies which involve entities and their relationships in different domains.

1.2 STRUCTURE OF THE SEMANTIC WEB

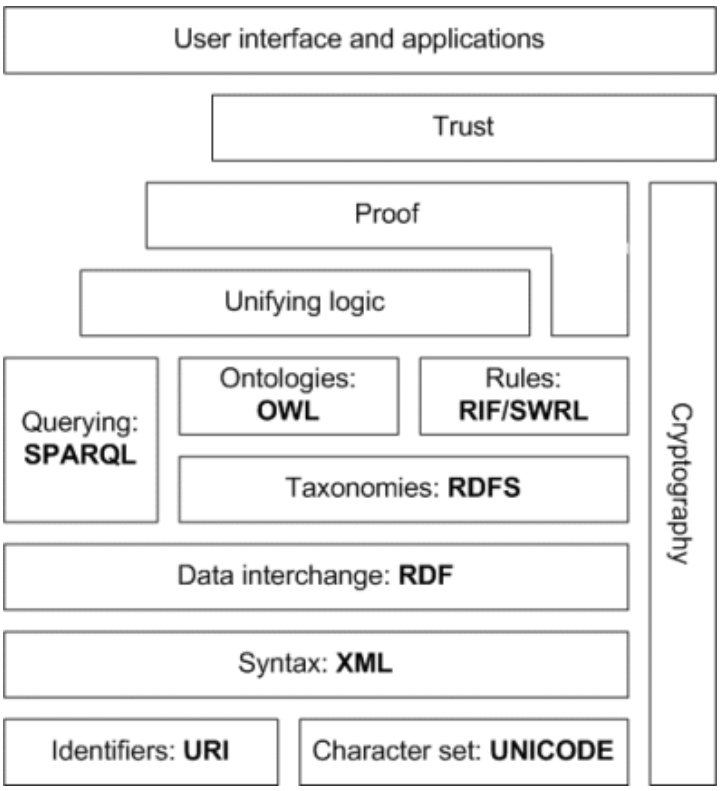

Figure 1.1 The Semantic Web Stack [3].

The figure above illustrates the Semantic Web layer cake. It is in the form of layers where each layer uses the capabilities of the layer below. The technologies from the bottom of the stack up to OWL are currently standardized and accepted to build Semantic Web applications [3].

The bottom layers contain technologies that are well known from the hypertext Web and that without change provide basis for the Semantic Web.

- Internationalized Resource Identifier (IRI), generalization of URI, provides means for uniquely identifying Semantic Web resources. Semantic Web needs unique identification to allow provable manipulation with resources in the top layers. A URI is simply a Web identifier: like the strings starting with "http:" or "ftp:" that are often found on the World

Wide Web. The World Wide Web is such a thing: anything that has a URI is considered to be "on the Web".

- Unicode serves to represent and manipulate text in many languages. Semantic Web should also help to bridge documents in different human languages, so it should be able to represent them.
- XML is a markup language that enables creation of documents composed of structured data. Semantic Web gives meaning (semantics) to structured data.
- XML Namespaces provides a way to use markups from more sources. Semantic Web is about connecting data together, and so it is needed to refer more sources in one document.

Middle layers contain technologies standardized by W3C to enable building Semantic Web applications.

- RDF is a standard model for data interchange on the Web [4]. It is a framework for describing resources on the Web. It is designed to be readable by computers and has an XML format
- RDF Schema (RDFS) provides basic vocabulary for RDF. Using RDFS it is for example possible to create hierarchies of classes and properties.
- Web Ontology Language (OWL) extends RDFS by adding more advanced constructs to describe semantics of RDF statements. It allows stating additional constraints, such as for example cardinality, restrictions of values, or characteristics of properties such as transitivity. It is based on description logic and so brings the reasoning power to the Semantic Web.
- SPARQL is an RDF query language - it can be used to query any RDF-based data (i.e., including

statements involving RDFS and OWL). Querying language is necessary to retrieve information for Semantic Web applications.

Top layers contain technologies that contain just ideas that should be implemented in order to fully realize the Semantic Web.

- RIF or SWRL will bring support of rules, for example to allow describing relations that cannot be directly described using description logic used in OWL.
- Cryptography is important to ensure and verify that Semantic Web statements are coming from trusted source. This can be achieved by appropriate digital signature of RDF statements.
- User interface is the final layer that will enable humans to use Semantic Web applications [3].

CHAPTER 2

BACKGROUND

2.1 ONTOLOGY: BASICS

An ontology is a specification of a conceptualization [4]. An ontology together with a set of individual instances of classes constitutes a knowledge base. Structurally, an ontology is a graph whose nodes represent concepts and arcs represent relationships between concepts.

An ontology typically provides a vocabulary that describes a domain of interest. Ontologies have been recognized as a crucial component for knowledge sharing and the realization of this vision. Two search strings with the same syntax can have dramatically different semantics. A user might be looking for pages about a less popular topic which happens to share the same syntax as a highly popular yet totally irrelevant topic. Finding what is needed will take time and ingenuity. This demonstrates the dire need for semantically enabled search engines. Ontologies provide a means for disambiguating syntactically equal but semantically different items. Ontologies tell the user that there are in fact different concepts, which all share the same language term. Ontologies are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. Ontologies are part of the W3C standards stack for the Semantic Web, in which they are used to specify standard conceptual vocabularies in which to exchange data among systems, provide services for answering queries, publish reusable knowledge bases, and offer services to facilitate interoperability across multiple, heterogeneous systems and databases.

The Working Group identified various ontology application categories which include web site or document organization and navigation support, browsing support, search support (semantic search), generalization or specialization of search, sense "disambiguation" support, consistency checking (use of restrictions), auto-completion, interoperability support (information/process integration), support validation and verification testing, configuration support and support for structured, comparative, and customized search.

As the scale of the World Wide Web has grown enormously, the demand for knowledge has been increasing rapidly. Search engines play an important role in providing information required from the Web these days. Similarly, the growth of the Semantic Web will also need a search mechanism which retrieves knowledge in the form of Semantic Web documents encoded in Semantic Web Lanugages like RDF and OWL. It is hard to navigate within the Semantic Web since few explicit ``hyperlinks" are available besides a URIref's namespace or owl:import [5].

Currently, the Semantic Web provides services such as searching Semantic Web ontologies, searching Semantic Web instance data, searching Semantic Web terms, i.e., URIs that have been defined as classes and properties and to provide metadata of Semantic Web documents and support browsing the Semantic Web archive different versions of Semantic Web documents.

The basic steps in developing an ontology are: defining classes in the ontology, arranging the classes in a taxonomic (subclass–superclass) hierarchy, defining properties and describing allowed values for these properties and filling in the values for instances.

Some of the needs for developing an ontology include sharing common understanding of the structure of information among people or software agents, enable reuse of domain knowledge, make domain assumptions explicit, separate domain knowledge from the operational knowledge and analyze domain knowledge.

Common components of ontologies include:

- Individuals: instances or objects (the basic or "ground level" objects).
- Classes: sets, collections, concepts, classes in programming, types of objects, or kinds of things.
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have.
- Relations: ways in which classes and individuals can be related to one another.
- Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement.
- Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input
- Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form.
- Axioms: assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only statements asserted as a priori knowledge. As used here, "axioms" also include the theory derived from axiomatic statements.
- Events: the changing of attributes or relations.

Ontologies are commonly encoded using ontology languages. As the computing world is moving towards the Semantic Web, people are interested in ontologies which form the backbone of the Semantic Web. With more and more ontologies coming into existence, ontology evaluation is gaining importance thus, allowing the developer to discover areas of improvements, to

understand the faults with the ontology created and to compare with other ontologies in the domain in order to understand how well it describes the domain semantically.

2.2 EXTENSIBLE MARK UP LANGUAGE (XML)

XML is a markup language for documents containing structured information. It is a set of rules for encoding information in a machine readable format. XML's design goals emphasize simplicity, generality, and usability over the Internet. XML uses tags for describing data. These tags are not predefined but have to be created and hence XML is said to be self descriptive. XML was designed to simplify data storage and sharing. It is used in many aspects of sharing data on the Web.

Some advantages of XML:

- XML makes data available easily since it is machine readable.
- XML data is stored in plain text format and hence it simplifies data sharing among different applications.
- XML simplifies working platform changes when upgrading to new systems. Since XML is stored in plain text format, it becomes easier to upgrade to new systems without incompatibility issues.
- XML simplifies data transport since it can be easily shared among incompatible applications.
- XML forms the basis of many new internet languages like XHTML, RDF, WSDL etc.

An example XML document would look like:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>Tim</to>
```

```
    <from>Alice</from>
    <heading>Invitation</heading>
    <body>Birthday party this weekend!</body>
</note>
```

The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set). The next line describes the root element of the document. The next four lines describe four child elements of the root (to, from, heading, and body). And finally the last line defines the end of the root element. It is very clear from the above example that the XML document describes a note from Tim to Alice for a birthday invitation.

2.3 RESOURCE DESCRIPTION FRAMEWORK (RDF)

RDF is a standard model for data interchange on the Web [6]. It is a framework for describing resources on the Web. It is designed to be readable by computers and has an XML format. It is a framework for creating statements in a form of so-called triples and enables to represent information about resources in the form of graph. By using XML, RDF information can easily be exchanged between different types of computers using different types of operating systems and application languages. The fundamental model of RDF is independent of XML. RDF is a model describing qualified (or named) relationships between two (Web) resources, or between a Web resource and a literal. The advantage of using RDF is that information is mapped directly and unambiguously to a model, a model which is decentralized, and for which there are many generic parsers already available. At that fundamental level, the only commonality between RDF and the XML World is the usage of the XML Schema data types to characterize literals in RDF.

Use of RDF:

RDF is a framework for describing web resources. For example, it can be used for describing properties for shopping items, such as price and availability, for describing time schedules for web events, for describing information about web pages (content, author, created and modified date), for describing content and rating for web pictures, for describing content for search engines, for describing electronic libraries.

RDF Model: The RDF model consists of three object types: Resources, Properties and Statements.

1. Resources - Resources are things being described by RDF expressions and are always named by URIs. A resource can be any HTML Document, specific XML element within the document source, collection of pages or a book.

2. Properties - Specific aspect, characteristic, attribute or relation used to describe a resource

For example, a book can be a resource with properties such as book_name, creator and title.

3. Statements - The combination of a Resource, a Property, and a Property value forms a Statement (known as the subject, predicate and object of a Statement).

RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”) [6].

An RDF triple contains three components

- the subject, which is an RDF URI reference or a blank node
- the predicate, which is an RDF URI reference
- the object, which is an RDF URI reference, a literal or a blank node

For example,

Arpan is the creator of the web page http://www.cs.uga.edu/~arpan/introduction/

The Subject (Resource) of the above statement is - http://www.cs.uga.edu/~arpan/introduction/

The Predicate (Property) of the resource is – Creator

The Object (Literal) is - Arpan

An RDF graph is a set of triples. The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph.

For the above example, the RDF graph would be:

Figure 2.1 A Simple RDF Graph

Consider the example below [6]:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">
<rdf:Description rdf:about="http://www.w3schools.com">
  <si:title>W3Schools.com</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>
</rdf:RDF>
```

Properties as Attributes:

The property elements can also be defined as attributes (instead of elements) as shown below [6]:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">
<rdf:Description rdf:about="http://www.w3schools.com">
  <si:title = “W3Schools.com” si:author = “Jan Egil
Refsnes”/>
</rdf:RDF>
```

The main elements of RDF are the root element, <RDF>, and the <Description> element, which identifies a resource. The first line of the RDF document is the XML declaration. The XML declaration is followed by the root element of RDF documents: <rdf:RDF>.

The xmlns:rdf namespace, specifies that elements with the rdf prefix are from the namespace. The xmlns:si namespace, specifies that elements with the si prefix are from the namespace http://www.w3schools.com/rdf/. The <rdf:Description> element contains the description of the resource identified by the rdf:about attribute. The elements <si:title> and <si:author> are properties of the resource.

Properties as Resources:

The property elements can also be defined as resources

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="http://www.w3schools.com/rdf/">
```

```
<rdf:Description rdf:about="http://www.w3schools.com">
 <si:title rdf:resource=”http://www.w3schools.com/rdf
/W3Schools”/>
 <si:author
rdf:resource=“http://www.w3schools.com/rdf/refsnes”/>
</rdf Description>
</rdf:RDF>
```

2.4 RDF Schema (RDFS)

Application-specific classes and properties must be defined using extensions to RDF. One such extension is RDF Schema [6]. The RDFS vocabulary is based on the vocabulary of RDF.

Classes: Resources can be divided into groups called classes. The members of a class are called instances. Classes are themselves resources. RDFS defines classes by using the following specifications [6]:

- rdfs:Resource is the class of everything. All things described by RDF are resources.
- rdfs:Class declares a resource as a class for other resources.
- rdfs:Literal is the class of literal values such as strings and integers. Property values such as textual strings are examples of RDF literals. Literals may be plain or typed. A typed literal is an instance of a datatype class.
- rdfs:Datatype is the class of all datatypes. Each instance of rdfs:Datatype is a subclass of rdfs:Literal.
- rdf:XMLLiteral is the class of XMLLiteral values. rdf:XMLLiteral is an instance of rdfs:Datatype (and thus a subclass of rdfs:Literal).
- rdf:Property is the class of RDF properties. rdf:Property is an instance of rdfs:Class.

Properties: Properties are relationships between subject resources and object resources. RDFS defines properties by using the following specifications [6]:

- rdfs:domain of an rdf:predicate declares the class of the subject in a triple whose second component is the predicate.
- rdfs:range of an rdf:predicate declares the class or datatype of the object in a triple whose second component is the predicate.
- rdf:type is a property used to state that a resource is an instance of a class.
- rdfs:subClassOf allows to declare hierarchies of classes.
- rdfs:subPropertyOf is an instance of rdf:Property that is used to state that all resources related by one property are also related by another.
- rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a resource's name.
- rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of a resource.

The following example demonstrates RDFS facilities:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">
<rdf:Description rdf:ID="animal">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
```

```
</rdf:Description>
<rdf:Description rdf:ID="horse">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
</rdf:RDF>
```

2.5 WEB ONTOLOGY LANGUAGE (OWL)

The Web Ontology Language is a knowledge representation language used to create ontologies. It is usually represented as RDF and is used to process information on the Web. It has an XML format and was designed to be readable by computers. OWL uses both URIs for naming and the description framework for the Web provided by RDF.

OWL ontologies can then be stored as documents in the World Wide Web. One aspect of OWL, the importing of ontologies, depends on this ability to store OWL ontologies in the Web. Owl extends RDFS to allow for the expression of complex relationships between different RDFS classes and of more precise constraints on specific classes and properties.

OWL Syntax:

An OWL ontology is an RDF graph (RDF Concepts), which is in turn a set of RDF triples. Thus, it is allowable to use other syntactic RDF/XML forms, as long as these result in the same underlying set of RDF triples.

Consider the following RDF/XML syntax:

```
<owl:Class rdf:ID="Person"/>
```

The following RDF/XML syntax encodes the same set of RDF triples, and therefore would convey the same meaning:

```
<rdf:Description rdf:about="#Person">
<rdf:type
rdf:resource="http://www.somewebsite.org/owl#Class"/>
</rdf:Description>
```

OWL has three sublanguages: OWL Lite, OWL DL and OWL Full.

OWL Lite is a sublanguage of OWL DL that supports only a subset of the OWL language constructs. It was originally intended to support those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. OWL Lite forbids the use of owl:one of, owl:unionOf, owl:complementOf, owl:hasValue, owl:disjointWith, owl:DataRange. Development of OWL Lite tools has thus proven almost as difficult as development of tools for OWL DL, and OWL Lite is not widely used.

OWL DL was designed to provide the maximum expressiveness possible while retaining computational completeness. It requires a pairwise separation between classes, datatypes, datatype properties, object properties, annotation properties, ontology properties, individuals, data values and the built-in vocabulary. This means that, for example, a class cannot be at the same time an individual. In OWL DL the set of object properties and datatype properties are disjoint. The following four property characteristics: inverse of, inverse functional, symmetric, transitive can never be specified for datatype properties. OWL DL requires that no cardinality constraints (local nor global) can be placed on transitive properties or their inverses or any of their superproperties. Annotations are allowed only under certain conditions. Most RDF(S) vocabulary cannot be used within OWL DL. All axioms must be well-formed, with no missing or extra components, and must form a tree-like structure [9].

OWL Full is based on a different semantics from OWL Lite or OWL DL, and was designed to preserve some compatibility with RDF Schema. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right; this is not permitted in OWL DL.

Each of these sublanguages is a syntactic extension of its simpler predecessor. The following set of relations hold. Their inverses do not [10].

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion**.**

2.6 THE STRUCTURE OF OWL ONTOLOGY

OWL is an important component of the Semantic Web activity. It aims to make Web resources more readily accessible to automated processes by adding information about the resources that describe or provide Web content. As the Semantic Web is inherently distributed, OWL must allow for information to be gathered from distributed sources. This is partly done by allowing ontologies to be related, including explicitly importing information from other ontologies. An OWL ontology consists of the following parts:

1. Namespaces

A standard initial component of an ontology includes a set of XML namespace declarations enclosed in an opening rdf:RDF tag. These provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation much more readable. For example,

```
<rdf:RDF
    xmlns="http://www.w3.org/TR/2004/REC-owl-guide-
20040210/wine#"
```

```
    xml:base="http://www.w3.org/TR/2004/REC-owl-guide-
20040210/wine#"
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

2. Ontology headers

After the namespaces, there is a collection of assertions about the ontology grouped under an owl:Ontology tag. These tags support such initial tasks as comments, version control and inclusion of other ontologies. For example,

```
<owl:Ontology rdf:about="">
<rdfs:comment>An example OWL ontology</rdfs:comment>
<owl:priorVersion rdf:resource="http://....."/>
<owl:imports rdf:resource="http://www....."/>
<rdfs:label>Some Ontology</rdfs:label>
  ...
```

Basic elements used in OWL [9]

Class: A class defines a group of individuals that belong together because they share some properties.

OWL distinguishes six types of class descriptions:

1. A class identifier (a URI reference): The first type describes a class through a class name (syntactically represented as a URI reference). For example, <owl:Class rdf:ID="Human"/>
2. An exhaustive enumeration of individuals that together form the instances of a class: The second type describes a class that contains exactly the enumerated individuals. The class

extension of a class described with owl:oneOf contains exactly the enumerated individuals, no more, no less. The list of individuals is typically represented with the help of the RDF construct rdf:parseType="Collection", which allows for writing down a set of list elements. For example,

```
<owl:Class
rdf:about="http://annotation.semanticweb.org/2004/places#Name">
<owl:oneOf rdf:parseType="Collection">
<owl:Thing rdf:about="#Georgia"/>
<owl:Thing rdf:about="#Florida"/>
<owl:Thing rdf:about="#Tennessee"/>
</owl:oneOf>
</owl:Class>
```

3. A property restriction: An owl:Restriction descriptor describes an anonymous class, namely a class of all individuals that satisfy the restriction. OWL distinguishes two kinds of property restrictions: value constraints and cardinality constraints. For example,

```
<owl:Restriction>
<owl:onProperty rdf:resource="(some property)" />
        (precisely one value or cardinality constraint)
</owl:Restriction>
```

A value constraint puts constraints on the range of the property when applied to this particular class description and a cardinality constraint puts constraints on the number of values a property can take. The value constraints are: owl:allValuesFrom, owl:someValuesFrom, owl:hasValue. The cardinality constraints are owl:maxCardinality, owl:minCardinality, owl:cardinality.

4. The intersection of two or more class descriptions: An owl:intersectionOf descriptor can be viewed as representing the AND operator on classes. In the example below, the value of owl:intersectionOf is a list of two class descriptions, namely two enumerations, both describing a class with two individuals. The resulting intersection is a class with one individual, namely Andy as this is the only individual that is common to both enumerations. For example,

```
<owl:Class
rdf:about="http://annotation.semanticweb.org/2004/names#Fname">
       <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
           <owl:oneOf rdf:parseType="Collection">
             <owl:Thing rdf:about="#Andy" />
             <owl:Thing rdf:about="#Jenny" />
           </owl:oneOf>
              </owl:Class>
              <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                  <owl:Thing rdf:about="#Candice" />
                  <owl:Thing rdf:about="#Andy" />
                </owl:oneOf>
              </owl:Class>
            </owl:intersectionOf>
</owl:Class>
```

5. The union of two or more class descriptions: An owl:unionOf descriptor can be viewed as representing the OR operator on classes. . In the example below, owl:unionOf describes an anonymous class for which the class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the list.. The resulting union is a class with three individuals, namely Andy, Jeny, Candice.

```
<owl:Class
rdf:about="http://annotation.semanticweb.org/2004/names#Fname">
            <owl:unionOf rdf:parseType="Collection">
              <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                  <owl:Thing rdf:about="#Andy" />
                  <owl:Thing rdf:about="#Jenny" />
                </owl:oneOf>
              </owl:Class>
              <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                  <owl:Thing rdf:about="#Candice" />
                  <owl:Thing rdf:about="#Andy" />
                </owl:oneOf>
              </owl:Class>
            </owl:unionOf>
</owl:Class>
```

6. The complement of a class description: An owl:complementOf descriptor can be viewed as representing the NOT operator on classes. For example,

```
<owl:Class>
  <owl:complementOf>
    <owl:Class rdf:about="#Vegetables"/>
  </owl:complementOf>
</owl:Class>
```

This description contains all the individuals that do not belong to the class Vegetables.

Individuals: Individuals are instances of classes, and properties may be used to relate one individual to another. Individuals are defined with axioms. Many axioms typically are statements indicating class membership of individuals and property values of individuals. For example, consider the following set of statements about an instance of the class Movie:

```
<Movie rdf:ID="Alice_In_Wonderland ">
  <hasDirector rdf:resource="#Tim_Burton"/>
  <hasActor rdf:resource="#Mia_Wasikowska"/>
  <hasActor rdf:resource="#Johnny_Depp"/>
  <hasActor rdf:resource="#Anne_Hathaway"/>
  <premiereDate rdf:datatype="&xsd;date">2010-03-
05</premiereDate>
  <premierePlace rdf:resource="#USA"/>
  <numberOfActs
rdf:datatype="&xsd;positiveInteger">4</numberOfActs>
</Movie>
```

OWL provides three constructs for stating axioms about individual identity

- owl:sameAs is used to state that two URI references refer to the same individual.
- owl:differentFrom is used to state that two URI references refer to different individuals

- owl:AllDifferent provides an idiom for stating that a list of individuals are all different.

Property: Property is a binary relation that states relationships between individuals or from an individual to data value. Property can be further distinguished as "ObjectTypeProperty" or "DataTypeProperty".

- ObjectTypeProperty (owl:ObjectTypeProperty): It is defined as the relation between instances of two classes
- DataTypeProperty (owl:DataTypeProperty) : It is defined as the relation between instances of classes and literal values such as string, number, and date.

For example,

```
<owl:ObjectProperty rdf:ID="hasParent"/>
```

Often, property axioms define additional characteristics of properties. OWL supports the following constructs for property axioms:

- RDF Schema constructs: rdfs:subPropertyOf, rdfs:domain and rdfs:range
- relations to other properties: owl:equivalentProperty and owl:inverseOf
- global cardinality constraints: owl:FunctionalProperty and wl:InverseFunctionalProperty
- logical property characteristics: owl:SymmetricProperty and owl:TransitiveProperty

Semantic Web research efforts are tackling the problem of querying semantically annotated documents. However, in the real world of WWW users the most popular search engines are still keyword-based. Even semantic search engines such as Swoogle [21], perform a term-based search in their repository, in order to retrieve Web ontology documents which contain concepts lexicalized by the query-term(s). Swoogle employs a system of crawlers to discover RDF documents and HTML documents with embedded RDF content. Swoogle uses the

page rank algorithm to search ontologies and does not perform a semantic search. Hence, there is a need for semantic search of ontologies.

CHAPTER 3

ONTOLOGY ALIGNMENT

3.1 WHAT IS ONTOLOGY ALIGNMENT ?

There are various sources of knowledge in Computer Science. Knowledge itself can be represented in various formats such as databases, collections of documents, files, emails, web pages, web services and so on. As the resources of knowledge are becoming cheaper, the flow of knowledge is increasing day by day. People are looking for means of gathering all this knowledge together and not only individual pieces of knowledge. Due to this, knowledge integration has always been a topic of continued attention and has continuously triggered further research and development. With the advent of the Internet and continuously evolving tools for data gathering and manipulation, physical data exchange is not an issue anymore. However, knowledge is represented in various formats and not all of them might be compatible with each other. One of the knowledge representations used widely is XML. However, it requires a lot of human effort to understand the semantics of data exchanged using XML. Identifying correspondence between different representations of knowledge belonging to the same domain is a difficult human task, almost impossible for machine based approaches at this time. The next step towards a better representation and understanding of knowledge is through an ontology. As the computing world is moving towards Semantic Web, people are interested in ontologies which form the backbone of the Semantic Web.

As mentioned in the previous chapter, an ontology provides a vocabulary to describe a domain of interest. There are various ontologies that describe various domains of knowledge.

Also, there are various ontologies that describe the same or very similar domain of knowledge as well. The Semantic Web community faces a problem when using these ontologies to represent knowledge which is, heterogeneity. Many ontologies have emerged recently, some of them representing the same contents. Despite of the same syntax, which is RDF or Web Ontology Language (OWL), these ontologies differ in the structure and nomenclature. The Semantic Web envisions making Web content machine processable, not just readable or consumable by human beings. This is accomplished by the use of ontologies which involve agreed upon terms and their relationships in different domains.

Ontology Alignment or Ontology Matching is the process of determining relationships between individual elements of different ontologies to understand how they interoperate. Historically, the need for ontology alignment arose out of the need to integrate heterogeneous databases, ones developed independently and thus each having their own data vocabulary. For computer scientists, concepts are expressed as labels for data. Ontology Alignment can be defined as the process of determining correspondences between concepts. The resulting set of corresponding concepts is known as the alignment. Finding alignment between onotologies can be useful to users for searching and browsing information from a variety of knowledge sources in a transparent way. Thus, the user can gain new insights by inferring from the multiple ontologies under consideration.

In the Semantic Web context involving many actors providing their own ontologies, ontology matching has taken a critical place for helping heterogeneous resources to interoperate. Ontology alignment tools find classes of data that are "semantically equivalent". A number of tools and frameworks have been developed for aligning ontologies. The overall goal of Ontology

alignment is to provide a concise methodology and implementation for aligning ontologies with each other.

The Matching Process

The Matching Process [13] determines the alignment A' for a pair of ontologies o and o'. There are some other parameters that extend this definition, namely: (i) the use of an input alignment A which is to be completed by the process (ii) the matching parameters p, e.g. weights, thresholds etc. (iii) external resources used by the matching process, r, e.g. common knowledge and domain specific thesauri.

Technically, this process can be defined as follows:

Formal Definition:

The Ontology Alignment process [13] can be seen as a function f, which, from a pair of ontologies to match, o and o', an input alignment A, a set of parameters p, a set of resources r, returns an alignment A' between these ontologies:

$$A' = f(o, o', A, p, r)$$

This can be schematically represented as:

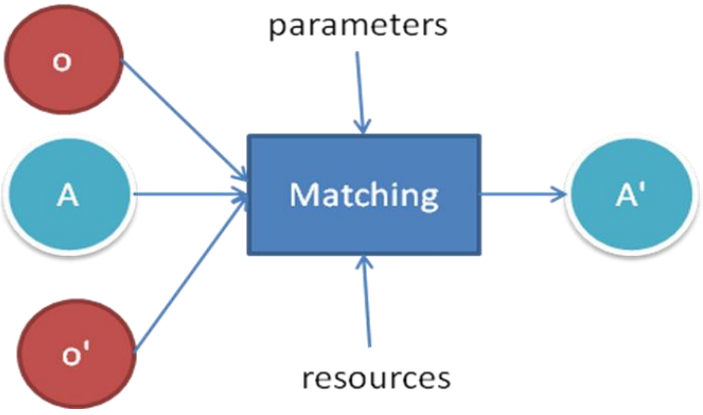

Figure 3.1 Ontology Matching Process

Ontology Alignment

Given two ontologies o and o′, an alignment is made up of a set of correspondences between pairs of entities belonging to QL(o) and QL ′(o′) respectively [13].

Alignment is the output of the matching process.

Challenges involved in Ontology Alignment

- Dealing with the real world with specific requirements.
- Data is of varying sizes from different sources.
- The semantic representation might not be complete and perfect.
- Alignment methods need to be flexible enough to be transferred to different domains, applications.

There are different algorithms to compute alignment between ontologies. These algorithms can be compared and/or classified based on the inputs they use to produce the result, the method used to determine the alignment and the output or the result produced

The algorithms can be classified according to the data model in which the ontologies are expressed. Another factor which can be used to compare alignment algorithms is the type of data used. For example, some algorithms use schema while others may use instance level information or some of them might even use both. Some algorithms discard information related to datatypes and use only the property names. Most of the algorithms focus on the class labels (i.e, concept names), their internal relationships as well as relationships with the neighboring concepts. The method of determining alignment between ontologies can either be an approximation (such as probability) or even some exact value determining the similarity measure. Element level matching is done by finding correspondences between elements of the ontologies in consideration and structure level alignment computes the mapping by analyzing how entities

appear together as a structure. Some algorithms interpret the input based on an already existing algorithm and some of them use external resources such as human input or thesaurus or any other existing knowledge base to process the given input.

3.2 ONTOLOGY ALIGNMENT EXAMPLE

The following example [11] illustrates alignments. The example consists of two simple ontologies that are to be aligned. The two ontologies O_1 and O_2 describing the domain of cars are given in Figure 3.2 [11]. The first ontology contains the six concepts object, vehicle, owner, boat, car, and speed, the two relations of belonging to somebody and speed, and the three instances Marc, Porsche KA-123, and 300 km/h. There is a subsumption relation between object, vehicle, and boat, resp. car; a vehicle is an object, a boat is a vehicle, etc. Each vehicle belongs to an owner and each car has a specific speed. On instance level, the Porsche KA-123 belongs to Marc and has the speed 300 km/h.

The second ontology covers the same domain but is modeled slightly differently. Beneath an overall thing concept, there exists a vehicle, which in turn has the subclasses automobile, Volkswagen, and Porsche. Further, there is a motor and speed. The automobile has a Motor which in turn has a property speed. A specific Porsche, Marc's Porsche, with the fast Motorl23456 is also represented. Reasonable alignment confidence values between the two ontologies are given in Table 3.1.

Each line contains the two corresponding entities from ontology 1 and ontology 2. In Figure 3.2, alignments are represented by the shaded channels each linking two corresponding entities. Obviously, things and objects, the two vehicles, cars and automobiles, as well as the two speeds are the same. The relations of having a speed and property correspond to each other, as they both

refer to speed. In addition, the two instances Porsche KA-123 and Marc's Porsche are the same, which are both fast.

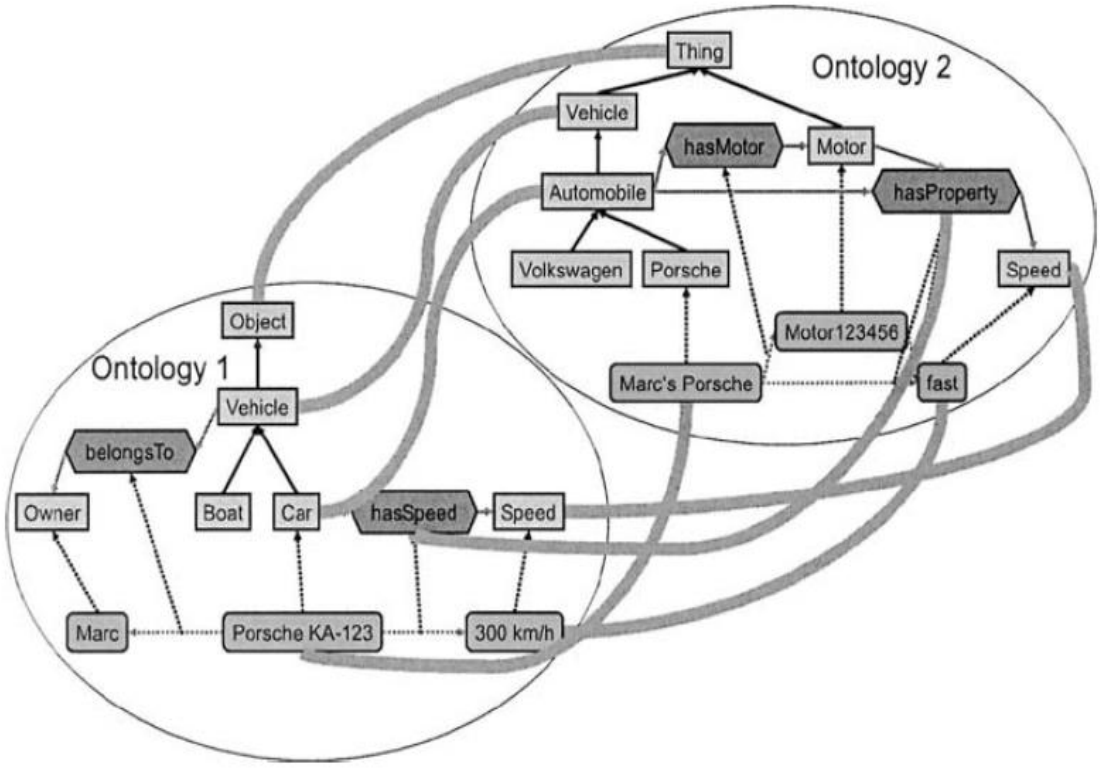

Figure 3.2 Ontology Alignment Example

Table 3.1 Confidence values between two ontologies to be aligned.

Ontology O_1	Ontology O_2	Confidence
object	thing	1.0
vehicle	vehicle	1.0
car	automobile	1.0
speed	speed	1.0
hasSpeed	hasProperty	1.0
Porsche KA-123	Marc's Porsche	1.0
300 km/h	fast	0.9

CHAPTER 4

ONTOLOGY CATEGORIZATION

4.1 INTRODUCTION

The goal of this work is to discover the domain of an unknown target ontology with respect to a large reference ontology (Wikipedia). The target ontology is categorized to a particular domain of knowledge based on the ontology alignment results. The tool developed to achieve this goal is named OntoClass.

Wikipedia has become quite useful and sometimes indispensable many web users. It is a free, online, shared-community, user-updatable, multi-lingual encyclopedia. It contains articles ranging from science to entertainment, from history to politics and just about any other domain one can possibly think of. Wikipedia being a collaborative effort of countless volunteers is always growing and expanding. Any new event, discovery, invention is almost immediately turned into a new Wikipedia article. Wikipedia has more than 3 million articles in the English language as of today. Around 1,200 articles are added to the Wikipedia knowledge base everyday as of August 2009 [14]. Hence, Wikipedia is highly up-to-date source of information. Having access to such a large source of information and using it only as an online encyclopedia seems like such a waste. Wikipedia, for the most part, contains information in weakly structured form i.e. in the form of continuous text, so it is not easily machine readable. It does contain information in the form of structured data in info-boxes and templates. Disambiguation pages are deceptively difficult to process automatically, since they are written in free text and often list items that are merely associated with the target term rather than senses of it. Page titles can also

present difficulties, because they often include additional scope information. However, that represents only a very small portion of the information available to us from Wikipedia. To collect information from unstructured data is not an easy task. Hence, we need to structure the information in such a way that it can be retrieved and made use of.

The DBpedia project [8] provides one way of structuring the Wikipedia data in the form of an ontology. Each article in Wikipedia is assigned a universal unique identifier which is nothing but a URI in ontological terms. Hence, each Wikipedia page is an entity in the ontology. Certain information about each entity is obtained using several extractors. For example, article titles are associated with a URI through the rdf:label property, disambiguation links are represented using the dbpedia:disambiguates property, etc. Each Wikipedia page contains an infobox which describes the most important information about the topic in question. This data or information is structured in nature and hence is represented using named links to the source page. Unstructured information also has connections to other information. However, they are represented as anonymous or href links with no meaning associated with them.

4.2 EXTERNAL MODULES USED BY OntoClass

4.2.1 WIKIPEDIA

The reference ontology that we use for OntoClass is also formed from Wikipedia. It is a slightly modified approach of the DBpedia ontology creation process mentioned above. The Wikipedia ontology utilized in this thesis was constructed by Maciej Janik in the LSDIS lab at UGA [19]. DBpedia concentrates more on infobox data extraction. Other types of templates present within the Wiki page do not receive any such special handling. In DBpedia, for each template present in the page, a new entity is created in the ontology though the template contains connections that are directly linked from the page. The Wikipedia ontology eliminates the

creation of these intermediate entities and creates direct connections instead. Separate property names have been created to distinguish between direct names, disambiguations and redirections. In Wikipedia articles, disambiguation is achieved by adding contextual information along with the name of the entity in parenthesis. For example, “ontology” and “ontology (information science)”. Phrases of this sort containing contextual information are not seen in documents since readers are capable of performing the disambiguation based on the document content. In the Wikipedia ontology, such names are shortened by removing the disambiguation information or contextual information and added as an alias name of a specific property to distinguish it from its full name.

4.2.2 WIKIPEDIA MINER TOOLKIT

The Wikipedia Miner toolkit [15] provides one way of navigating and making use of structure and content of Wikipedia. It aims to make it easy to integrate Wikipedia's [16] knowledge into applications, by:

- providing simplified, object-oriented access to Wikipedia's structure and content.
- measuring how terms and concepts in Wikipedia are connected to each other.
- detecting and disambiguating Wikipedia topics when they are mentioned in documents.

The Wikipedia Miner can detect Wikipedia topics when they are mentioned in documents. It uses the links found in Wikipedia's articles to identify different senses for terms. It has two useful features which can be used to obtain results:

The Search service allows a user to treat Wikipedia as a gigantic thesaurus, for describing everything. The Wikipedia articles this service locates provide a wide array of useful linguistic information, including definitions, synonyms, translations, and related topics. The search

vocabulary is extensive (5 million or more terms and phrases), and encodes both synonymy and polysemy.

The Compare service allows a user to compare terms and concepts to measure how strongly they relate to each other. For this work, the Search feature was used to obtain corresponding Wikipedia articles and categories. Wikipedia Miner also gives results in XML format which was parsed to obtain the required articles and categories.

4.2.3 WORDNET

OntoClass, the system described in this thesis, also uses WordNet [20], which is a large lexical database of English, Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

4.2.4 WIKIPEDIA CATEGORIZATION SERVER

OntoClass has another module called as Wikipedia Categorization Server [19]. It is a text categorization system which uses Wikipedia ontology to classify a given document containing text. The classification is done using a semantic graph which is nothing but a mapping between the input text and Wikipedia ontology. The input to the server is a document containing text and it returns output which is the thematic graph (in the form of XML) containing the nodes, edges and possible categories of the input text. The categorization algorithm used by Wikipedia Categorization Server consists of three steps: Construction of semantic graph, selection and analysis of the thematic graph and categorization of the selected thematic graph.

1. Semantic graph construction

- Named entities identification – Phrases describing entities (entity labels) in the ontology are matched in the text. For each located phrase, all associated entities are added as nodes to the created semantic graph. Each node is assigned an initial weight based on the strength of the match.
- Connectivity inducing – Edges between the nodes in the semantic graph are created based on the relationships existing in the ontology and connecting the entities corresponding to the nodes. Each edge is assigned a weight based on the importance of the relationship in the ontology schema.
- Information propagation – Node weights are propagated to their neighbors in order to establish the most authoritative entities in the graph.

2. Thematic graph selection and analysis

- Connected component identification – Connected components in the semantic graph are identified, treating the graph as undirected.
- Dominant thematic graph identification – The largest and most important connected component of the semantic graph is selected as the dominant thematic graph.
- Core selection – The most important and/or central entities in the thematic graph are identified; they form the core of the thematic graph.

3. Dominant thematic graph categorization

- Class assignment – Each entity in the dominant thematic graph is assigned a set of classes, according to the entity's classification in the ontology (assuming that an entity may belong to multiple classes). For each class in the set, its depth in the ontology class hierarchy is recorded.

- Ontological classification – Starting with the classes assigned to the authoritative and/or central entities, ascend in the ontological class hierarchy until a set of parent classes is located that covers a significant portion of entities in the dominant thematic graph. Each class in the new set of (higher level) classes is ranked according to (i) the weight of the entities it covers, (ii) the percentage of the covered entities in the dominant thematic graph, and (iii) the distance in the class hierarchy to the covered entities.
- Categorization – the target categories are defined as ontology class (sub-)hierarchies, or just lists of classes; each class in the set located in the previous step is determined to belong to one or more categories (as a member of the hierarchy or the list); the weight of the classes is used to determine the best category for the document.

4.3 OntoClass : A TOOL FOR ONTOLOGY CATEGORIZATION

OntoClass uses three methods to categorize an unknown target ontology. The first method uses target ontology class names to determine its category. The second method uses entire text (class names, labels, comments, properties) present in the target ontology to determine its category. The third method uses hierarchy of parents obtained from Wikipedia Miner to categorize the target ontology. The categorization methods used by OntoClass are as follows:

4.3.1 OntoClass with target ontology classes

The first categorization method uses target ontology class names to categorize the target ontology. The target ontology was cleaned up to remove any special characters and prefixes to retain all the class names in the document. The cleaned up document containing the target ontology classes was sent to the Wikipedia Categorization Server to which creates a mapping between the target and reference ontologies and returns a thematic graph in the form of XML document. The graph nodes contain the names of Wikipedia articles and the weights associated

with them. Once the thematic graph was obtained from the Wikipedia Categorization Server in the form of an XML document, the XML-DOM library along with JUNG was used to parse the XML and represent it in the form of a undirected graph. Various importance calculation algorithms such as Degree Centrality, Eigenvector Centrality, and Barry center score available in the JUNG package were run on the thematic graph obtained from the Wikipedia Categorization Server. These centrality values were made use of to find the most important entities in the thematic graph. Node scores were calculated for each node by adding the node weight and the sum of centrality values from each of the above mentioned algorithms. The nodes were arranged in descending order of node scores. If the number of nodes in the list was less than ten, the entire list was retained else first half of the list of nodes was added to the first list.

Each thematic graph node was compared with target ontology classes to find a match. If there was no match, a synonym of the thematic graph node was found using Wordnet and compared with the target ontology classes. The thematic graph nodes or their synonyms which matched the target ontology classes were added to the second list. The list of important nodes was obtained by combining the first and the second lists. The list of important nodes was sent to the categorizer to obtain the final categories.

4.3.2 OntoClass with target ontology classes and text

The second categorization method uses target ontology text (which includes class names, labels, comments, properties) to categorize the target ontology. The target ontology was cleaned up to remove any special characters and prefixes to retain all the text in the document. The cleaned up document was sent to the Wikipedia Categorization Server to which creates a mapping between the target and reference ontologies and returns a thematic graph in the form of XML document. The graph nodes contain the names of Wikipedia articles and the weights

associated with them. Once the thematic graph was obtained from the Wikipedia Categorization Server in the form of an XML document, the XML-DOM library along with JUNG was used to parse the XML and represent it in the form of a undirected graph. Various importance calculation algorithms such as Degree Centrality, Eigenvector Centrality, and Barry center score available in the JUNG package were run on the thematic graph obtained from the Wikipedia Categorization Server. These centrality values were made use of to find the most important entities in the thematic graph. Node scores were calculated for each node by adding the node weight and sum of the centrality values from each of the above mentioned algorithms. The nodes were arranged in descending order of node scores. If the number of nodes in the list was less than ten, the entire list was retained else first half of the list of nodes was added to the first list.

Each thematic graph node was compared with target ontology text to find a match. If there was no match, a synonym of the thematic graph node was found using Wordnet and compared with the target ontology classes. The thematic graph nodes or their synonyms which matched the target ontology classes were added to the second list. The list of important nodes was obtained by combining the first and the second lists. The list of important nodes was sent to the categorizer to obtain the final categories.

Calculation of centrality scores

For each node of the thematic graph, graph ranking measures such as Degree centrality, Eigen Vector centrality and Barry center score were determined using JUNG package. These scores determine the important nodes in the semantic graph and are known as centrality scores. There exist several centrality scoring algorithms. The measures mentioned below are some of the few we use in our system:

Degree centrality

Degree Centrality is defined as the number of edges connected to or incident on a node. In other words, degree of a node is the sum of the fan-in and fan-out values of the node. A node with a high degree centrality score implies that the entity is related to several other entities in the graph and hence is an important and popular entity within the graph. Most of the nodes in a semantic graph can be found in the target ontology on which it is based. Hence, they are all already connected to each other. The degree centrality of a node represents how well an entity is connected to the other entities in the ontology.

The Degree Centrality module uses the degree centrality scores of each of the nodes in the semantic graph to compute importance of nodes and ultimately the final dominant category of the unknown target ontology.

Eigenvector centrality

A more sophisticated version of the Degree Centrality measure is the Eigenvector centrality. Where degree centrality gives a simple count of the number of connections a vertex has, eigenvector centrality acknowledges that not all connections are equal. The eigenvector centrality of a node depends both on the number and the quality of its connections obtained by the average centrality values of the adjacent nodes. Eigenvector measure of a node is calculated using the principle that connections to high-scoring nodes contribute more to the score of the said node than low-scoring nodes. Hence, a node with a small number of heavy-weight connections may have a higher score when compared to a node with a large number of low-weight connections. This would also determine the important nodes and their connections in the semantic graph and ultimately the most dominant category.

Barry center centrality

The Distance Centrality score of vertices is calculated based on their distances to each and every other vertex in the graph. There are 2 types of distance centrality scores – Closeness Centrality and Barry center Centrality. Closeness of a vertex within a graph is higher if it has short geodesic distances to all other vertices in the graph. Geodesic distance is calculated as the number of edges in the shortest path connecting the two vertices. If there exists no path between the two vertices i.e. they belong to different connected components, then the geodesic distance between them is infinity.

Barry center scores are calculated as - 1 / (total distance from vertex v to all other vertices). Hence, Barry center scores are assigned to each vertex according to the sum of its distances to all other vertices. If the total sum of the distances to all other nodes in the graph is high, it implies that the vertex is a non-central entity in the graph, since that node is not directly connected to many other entities in the graph.

4.3.3 OntoClass with parent hierarchy from Wikipedia Miner Toolkit

The target ontology was cleaned up to remove any special characters and prefixes to retain all the class names. For each target ontology class in the cleaned up document, a list of corresponding matches with the commonness score values was obtained from Wikipedia Miner. The term with highest sense commonness was selected for each target ontology class and a list of all mapped reference ontology entities was obtained. For each mapped reference ontology entity, parent categories were found at the first and second levels. The second level list of parents was traversed to find the frequency of each parent term in the list. Those terms with a frequency of one were eliminated. The parent categories of the remaining terms were found and a third level parent list was created. The third level parent list was traversed to find the frequency of each

term and the term which covered almost all the classes was reported as the final category of the target ontology.

4.4 SYSTEM ARCHITECTURE

4.4.1 OntoClass with target ontology classes / target ontology classes and text

The first two methods are similar except for output obtained the first stage. In the first method, the target ontology parser provides a list of target ontology class names as the input to Wikipedia Categorization Server [18] whereas in the second method, it provides the ontology text (classes along with any phrases, labels, comments and properties) as the input to the Wikipedia Categorization Server. The system architecture is as follows:

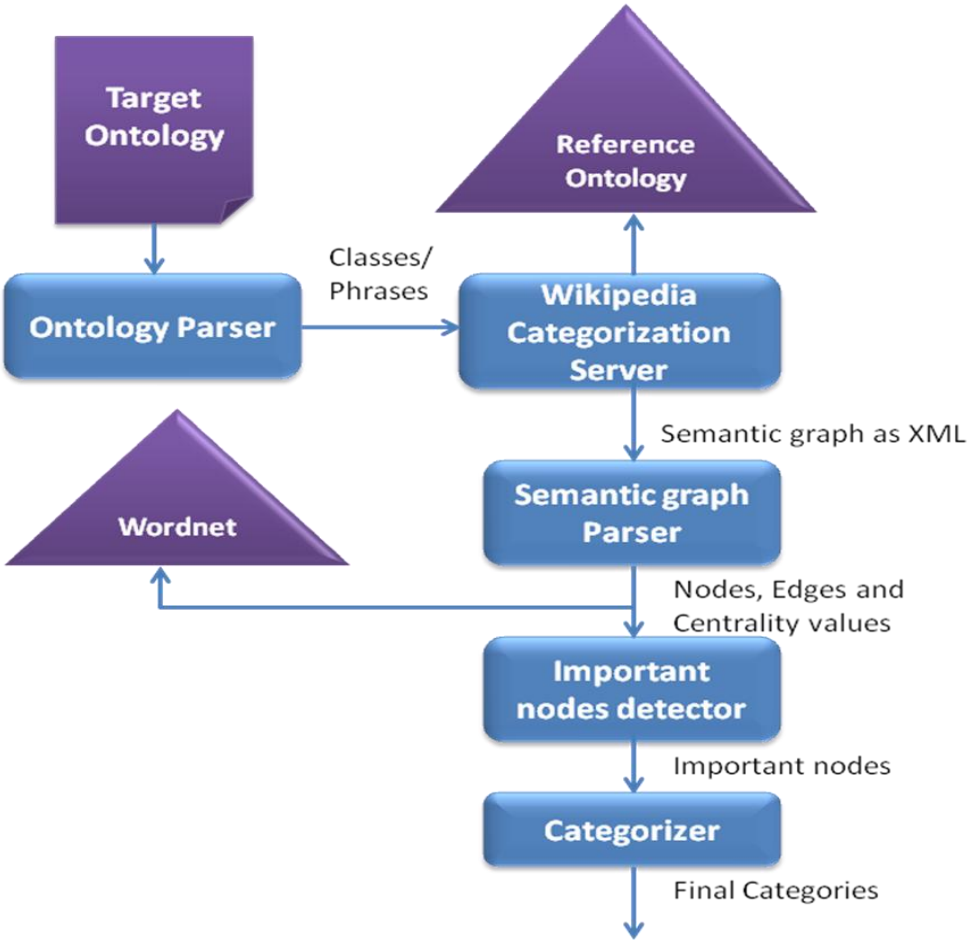

Figure 4.1 System Architecture (first two methods)

Input: The input to the system is the unknown target ontology to be classified.

Ontology Parser: The target ontology is cleaned up by the ontology parser module to remove any special characters and prefixes to retain all the class names (along with text for the second method) in the document. The cleaned up document containing the target ontology class names(along with text for the second method) is sent to the Wikipedia Categorization Server.

Wikipedia Categorization Server: This module helps in achieving correct classification by providing the semantic graph which is further traversed to find important and dominant categories. The input to the Wikipedia Categorization Server module is the list of target ontology classes (along with text for the second method) provided by the Ontology Parser. This module makes use of the Wikipedia ontology and creates a mapping between the ontology and the input which is nothing but the semantic graph. The output is the thematic graph in the form of an XML document.

Important nodes detector: This module detects the important nodes in the thematic graph and sends the list of important nodes to the categorizer. It generates the list of important nodes with the help of node scores as described in the methods in the previous section.

Categorizer [19]: The list of important nodes is sent to the categorizer to obtain the final categories.

4.4.2 OntoClass with Parent hierarchy provided by Wikipedia Miner Toolkit

Ontology Parser: The target ontology is cleaned up by the ontology parser module to remove any special characters and prefixes to retain all the classes. The cleaned up document containing the target ontology class names is sent to the Wikipedia Miner Parser.

Wikipedia Miner Parser: It consults the Wikipedia Miner toolkit and extracts a list of corresponding matches with the commonness score values for each target ontology class

obtained from the Ontology Parser. It then extracts the term with highest commonness value for every target ontology class to get a list of all corresponding reference ontology entities. This list is sent to the Parent Category Extractor.

Parent Category Extractor: For each mapped reference ontology entity, parent categories are found at the first and second levels. The second level list of parents is traversed to find the frequency of each parent term in the list. Those terms with a frequency of one is eliminated. The parent categories of the remaining terms were found and a third level parent list is created. The third level parent list is traversed to find the frequency of each term and the term which is dominant is reported as the final category of the target ontology.

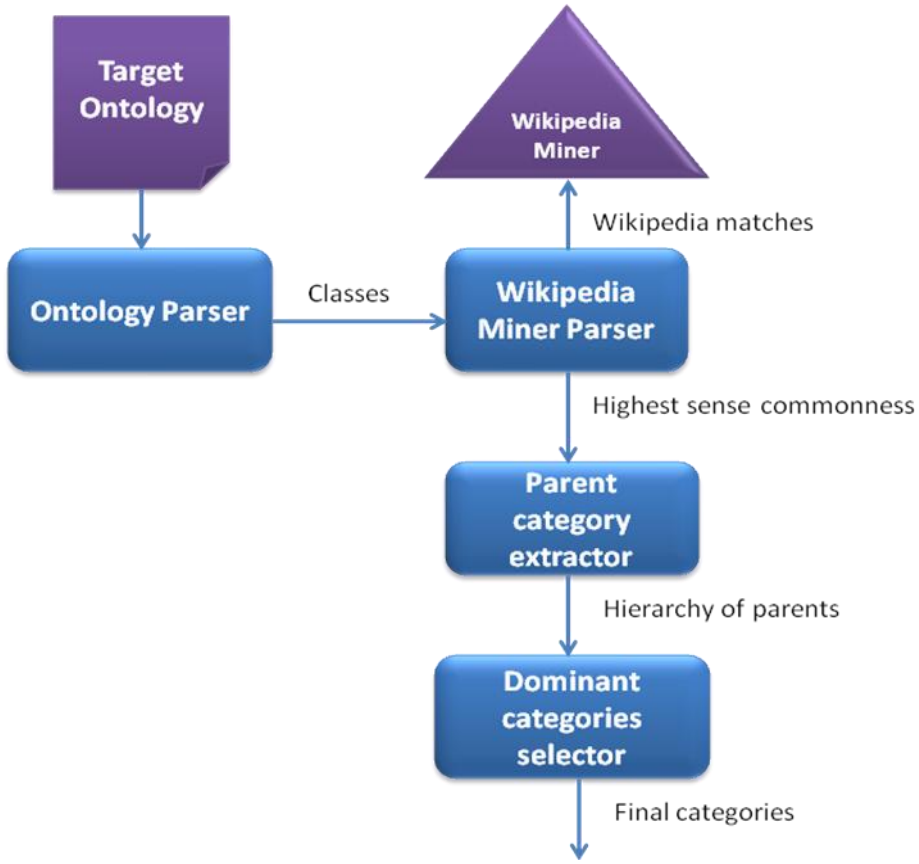

Figure 4.2 System Architecture (third method)

The system has been implemented in the Java programming language. Some of the libraries that were used: Java Universal Network/Graph Framework or JUNG [17] is a software library that provides a common and extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network. The distribution of JUNG includes implementations of a number of algorithms from graph theory, data mining, and social network analysis, such as routines for clustering, decomposition, optimization, random graph generation, statistical analysis, and calculation of network distances, flows, and importance measures such as degree centrality, Eigenvector centrality and Barry center score. We used the JUNG package for graph representation of the XML and to run algorithms to calculate the various importance measures for the graph.

CHAPTER 5

EXPERIMENTS AND EVALUATIONS

5.1 EXPERIMENT SETUP

The tool has been developed using Java programming language. Some of the libraries used were the Java Universal Network/Graph or the JUNG Framework. The Wikipedia Categorization Server [18] is hosted on the kronos server at UGA's LSDIS lab. We setup a testing environment to run experiments and evaluate our system modules. Experiments were conducted using twenty target ontologies belonging to different domains. Each of these ontologies was tested on the three methods described in the previous chapter. The input target ontologies were of different types in terms of size, formats and the domains they represented. The target ontologies were evaluated using certain tools as well as human judgement and how well they were categorized by the system.

5.2 EVALUATION METRICS

Evaluating such a system is difficult since there is no automatic means of evaluation. Hence, human judgement was used for evaluation. The target ontologies were evaluated based on their characteristics and metrics as described below. Accordingly, they have been classified as superficial, medium and elaborate.

The system was tested with twenty different ontologies belonging to various domains. Different characteristics for each target ontology, such as creator name, class count, object property count, data property count, individual count, isolated entity count, missing labels and comments count, missing inverse object property count are listed below. These characteristics were obtained by

using ontology editors such as Protégé [23] and Neon Toolkit [7]. Each input target ontology was evaluated on the basis of certain metrics [22] such as:

- Relationship richness (RR): The relationship richness of a schema is defined as the ratio of the number of relationships defined in the schema, divided by the sum of the number of subclasses.
- Attribute richness (AR): The attribute richness is defined as the average number of attributes (slots) per class. It is computed as the number attributes for all classes divided by the number of classes.
- Average population (AP): The average population of classes in a Knowledge Base is defined as the number of instances of the Knowledge Base divided by the number of classes defined in the ontology schema.
- Inheritance richness (IR): The inheritance richness of the schema is defined as the average number of subclasses per class.

  Each of these evaluation metrics were calculated for each target ontology used to test the system. The metric values for each ontology have been summarized in the table below.

  Table 5.1 Evaluation metrics for the target ontologies used.

Name	Class count	Object Prop. count	Data Prop. count	Missing Labels/ Comments	RR	AR	AP	IR	Missing Domain or Ranges	Ontology Type
Film	95	385	0	0	0.8	0	0	1	0	Elaborate
University	30	11	1	47/47	0.24	0.03	0.13	1.13	0	Medium
Restaurant	165	29	28	111/110	0.12	0.16	0	1.26	16	Elaborate

Name	Class count	Object Prop. count	Data Prop. count	Missing Labels/ Comments	RR	AR	AP	IR	Missing Domain or Ranges	Ontology Type
Olympics	146	26	14	128/128	0.14	0.09	0	0.95	15	Elaborate
Generations	18	4	0	30/30	1	0	0.38	0	7	Medium
Family	14	23	2	58/57	0.25	0.14	0.71	0.57	35	Superficial
ISWC	33	18	17	119/118	0.21	0.51	1.51	2.03	36	Medium
Lipid	716	46	0	762/761	0.08	0	0	0.70	25	Elaborate
Animals	53	2	5	54/53	0.03	0.15	0.60	0.98	0	Elaborate
Diseases	7810	7	0	7969/ 7969	0	0	0.01	1.64	0	Elaborate
Family Health	239	432	1	684/684	0.70	0	0.05	0.74	777	Elaborate
Finance	270	113	20	416/416	0.30	0.07	0.04	0.93	6	Elaborate
Life Science	34	4	0	164/163	1	0	3.79	0	4	Superficial
Airtravel	117	40	11	193/193	0.25	0.09	0.29	1	15	Elaborate
Geographical Regions	39	0	0	40/40	0	0	0	0.97	0	Superficial
Parasite	41	5	0	47/46	0.09	0	0	1.19	0	Medium
Book	4	8	3	16/16	0.88	0.75	0	0.25	0	Superficial

Name	Class count	Object Prop. count	Data Prop. count	Missing Labels/ Comments	RR	AR	AP	IR	Missing Domain or Ranges	Ontology Type
Baseball	78	20	1	16/16	0.16	0.01	1.16	1.30	0	Elaborate
Name	Class count	Object Prop. count	Data Prop. count	Missing Labels/ Comments	RR	AR	AP	IR	Missing Domain / Ranges	Ontology Type
Beer	58	11	4	70/70	0.19	0.06	0.15	0.77	5	Elaborate
Food	138	16	1	110/109	0.06	0	1.49	1.65	0	Elaborate

5.3 RESULTS

The system is evaluated by testing different target ontologies. The various ontologies used and the results obtained are summarized in the table below:

Table 5.2 OntoClass Results

Ontologies used	OntoClass with target ontology classes	OntoClass with target ontology text	OntoClass with Wikipedia Miner
Baseball	Baseball Baseball_terminology Baseball_rules Ball_and_bat_games Ball_games	Baseball Baseball_terminology Baseball_rules Olympics_sports Team_sports	Sports
Film (imdb)	Media_occupations Film_crew Entertainment_occupations Filmmakers Mass_media	Entertainment Arts Humanities Culture Society	Entertainment

University	Academia Education_and_training_occupations Education Titles Higher_education	Academia Education_and_training_occupations Titles Education Higher_Education	Society Social institutions Personal development Knowledge sharing Knowledge Fundamental Educational stages Articles Academia
Olympics	Olympic_sports Team_sports Sports Olympics Ball_games	Olympic_sports Sports Team_sports Games Ball_games	Sports
Generations	Family Human_development Kinship_and_descent Society Social_psychology	Family Human_development Kinship_and_descent Society Social_psychology	Interpersonal_relationships Society
Book	Books Written Communication Literature_by_medium Printing Publications _by_ format	Writing Communication,Human_skills Information_systems Language	No results since it is a very small ontology. All parents occur once…they get eliminated during the second level parsing.
Family	Human_development Humans Childhood Youth Time	Human_development Humans Time Biology Anthropology	Society
ISWC(conference) does not contain many terms specific to ISWC.	Academia Education_and_training_occupations Education Higher_education Titles	Academia Titles Education_and_training_occupations Education Higher_education	Society Education

Lipid	Organic_compounds Organic_chemistry Chemistry Chemical_compounds Carbon_compounds	Chemistry Organic_chemistry Organic_compounds Chemical_elements Chemical_substances	Chemistry Carboxylic acids Biomolecules Amines
Animals	Mammals Animals Vertebrates Parent_categories Tetrapods	Mammals Animals Vertebrates Parent_categories Tetrapods	Animals Fauna_by_continent
Diseases	Inflammations Medical_specialties Medicine Diseases Health	No results. Nineteen thousand lines of text generated.	Medical_specialities Biology
Family Health	Family Human_development Kinship_and_descent Society Social_psychology	Family Human_development Kinship_and_descent Society Social_psychology	Science
Finance	Stock_market Finance Business Economics Financial_economics	Stock_market Business Finance Financial_economics Main_topic_classifications	Finance Business
Lifescience	Molecular_biology Biology Genetics Life Biochemistry	Biology Molecular_biology Life Genetics Natural_sciences	Biology Natural_sciences Chemistry
Airtravel	Aviation_terminology Transportation Aviation Airfield Industries	Transportation Aviation Airfield Rail_transport Industries	Aviation Building_and_structures_by_type
GeographicalRegions	Continents Poles Americas Earth Places	Continents Earth Places Landforms Plate_tectonics	Countries_by_continents Countries

Parasite	Euglenozoa Parasitic_protists Parasitism Parasitology Symbiosis	Biology Life Natural_sciences Scientific_classification Parasitology	No results. Terms very specific to the domain
Beer	Beer_styles Beer Types_of_beer Fermented_beverages German_loanwords	Types_of_beer Beer Fermented_beverages Types Structure	Foods
Food	Meals Food_and_drink Foods Culture Digestive_system	Fish_products Seafood Foods Meat Animal_products	Food_and_Drink Foods

5.4 DISCUSSION

The Film ontology has ninety five classes which are well connected, with no isolated entities. It has no individuals. It has a high value for relationship richness which means that the relationships are not just IS-A relationships. There are no attributes and hence the attribute richness value is zero. Its classes are well distributed and cover most of the aspects of the domain. The first method classified this ontology moderately because majority of the classes define film crew occupations. The second method classified this ontology more accurately than the first one most likely due to the presence of other supporting information in the form of properties, labels and comments. The third method also classified the ontology more accurately than the first one.

The University ontology is a small ontology with thirty classes. It has four individuals. It is well connected but has a low relationship richness which means that there are more IS-A relationships. The attribute richness value is low, so there is not much information about each class in the ontology. The ontology describes the domain accurately because of its small size and

more specific classes. The first method classifies this ontology accurately because the classes of the target ontology describe the domain most accurately. The second method produced similar results to the first method due to the presence of text and phrases relevant to the domain. The third method gave moderate results.

The Restaurant ontology is a medium sized ontology with one hundred sixty five classes. It has no individuals. It is well connected and has two isolated entities. It has low relationship richness which means that there are more IS-A relationships. The attribute richness value is low, so there is not much information about each class in the ontology. The first method moderately classified the ontology because the ontology contains more terms related to food and drink than terms describing the restaurant itself. The second method also classified the target ontology moderately but better than the first method due to the existence of properties and comments which make the categorization better. The third method was the least accurate in this case.

The Olympics ontology is a medium sized ontology with one hundred forty six classes. It is also well connected with no isolated entities. It has no individuals. It has low relationship richness which means that there are more IS-A relationships. It has low attribute richness value. It is six levels deep. The classes and relationships describe the domain accurately. All the three methods classified the ontology accurately.

The Generations ontology is a small sized ontology with eighteen classes. It is well connected with no isolated entities. It has a depth of three levels. It has seven individuals. It has high relationship richness value because there are less number of IS-A relationships and low attribute value due to absence of class attributes. All the three methods classified the ontology moderately.

The Family ontology is a small sized ontology with fourteen classes. It is well connected with no isolated entities. It has a depth of three levels. It has ten individuals. It has low relationship

richness value because there are more number of IS-A relationships and low attribute value due to low number of class attributes. Even though the ontology has been named as describing a family, it does not capture the domain accurately. It lacks all the information needed to represent a family. As per the classes present in the target ontology, all the three methods classified it accurately.

The ISWC ontology is a small sized ontology with thirty three classes. It is well connected with no isolated entities. It is four levels deep. It has fifty individuals. It has low relationship richness value which means that there are more number of IS-A relationships. It has low attribute richness value. Even though, it is named as describing ISWC in specific, it does convey very little information about ISWC. It represents information in a more general way and has classes and relationships that describe a conference in general. All the three methods moderately classified the target ontology.

The Lipid ontology is of big size with seven hundred sixteen classes. It is well connected with no isolated entities. It has no individuals. It has low relationship richness and attribute richness values. The first two methods accurately classified the ontology. The third method produced a more general classification.

The Animals ontology is a small sized ontology with fifty three classes. It is well connected with no isolated entities. It has a depth of five levels. It has thirty two individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute richness value due to low number of class attributes. It describes the domain accurately with classes that represent most of the information about the domain. As per the classes present in the target ontology, all the three methods classified it accurately.

The Diseases ontology is a big sized ontology with seven thousand eight hundred ten classes. It is well connected with no isolated entities. It has a depth of three levels. It has one hundred fifty four individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute richness value due to low number of class attributes. The first method classified the target ontology accurately. The second method did not provide results due to huge size of text (nineteen thousand lines) obtained after parsing the target ontology. The third method did not classify the target ontology accurately. It produced a very broad classification.

The Family Health ontology is a medium sized ontology with two hundred thirty nine classes. It is well connected with no isolated entities. It has a depth of eight levels. It has twelve individuals. It has high relationship richness value because there are less number of IS-A relationships and low attribute richness value due absence of class attributes. All the three methods classified it moderately.

The Finance ontology is a medium sized ontology with two hundred seventy classes. It is well connected with one isolated entity. It has a depth of eight levels. It has twelve individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute richness value due to low number of class attributes. As per the classes present in the target ontology, all the three methods classified it accurately.

The Life Science ontology is a small sized ontology with thirty four classes. It is not at all well connected with thirty four isolated entities. It has a depth of two levels. It has one hundred twenty nine individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute richness value due to low number of class attributes. As the

ontology does not provide enough information about the domain, all the three methods classified it moderately.

The Air travel ontology is a medium sized ontology with one hundred seventeen classes. It is well connected with no isolated entities. It has a depth of six levels. It has twenty four individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute richness value due to low number of class attributes. It describes the domain accurately. All the three methods classified it accurately.

The Geographical Regions ontology is a small sized ontology with thirty nine classes. It is well connected with no isolated entities. It has a depth of four levels. It has no individuals. It has low relationship richness value because there are only IS-A relationships and low attribute richness value due absence of class attributes. In spite of having only IS-A relationships, it has very specific terms that describe the domain. All the three methods classified it accurately.

The Parasite ontology is a small sized ontology with forty one classes. It is well connected with no isolated entities. It has a depth of five levels. It has no individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute richness value due to absence of class attributes. The first two methods classified it accurately, but the third method did not produce any results due to very specific terms used in the ontology.

The Book ontology is a small sized ontology with four classes. It is well connected with no isolated entities. It has a depth of three levels. It has no individuals. It has high relationship richness value because there are less number of IS-A relationships and high attribute richness value due to absence of class attributes. The first method classified it accurately whereas the second method produced a slightly broader classification. The third method did not produce

results because the ontology is very small and the parents got eliminated at the second level itself.

The Baseball ontology is a small sized ontology with seventy eight classes. It is well connected with no isolated entities. It has a depth of six levels. It has ninety one individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute value due to absence of class attributes. It has good number of instances. The first two methods classified it accurately. The third method classified the ontology to a broader domain.

The Beer ontology is a small sized ontology with fifty eight classes. It is well connected with no isolated entities. It has a depth of six levels. It has nine individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute value due to absence of class attributes. The first two methods classified it accurately. The third method classified the ontology to a broader category.

The Food ontology is a medium sized ontology with one hundred thirty eight classes. It is well connected with no isolated entities. It has a depth of seven levels. It has two hundred six individuals. It has low relationship richness value because there are more number of IS-A relationships and low attribute value due to absence of class attributes. All three methods classified the target ontology accurately.

5.5 OVERALL EVALUATION OF THE SYSTEM

The system was evaluated with a numeric assessment to test its categorization quality. An overall evaluation was done by using a scoring method which is as follows:

- A score of 3 points was assigned to a method if the result produced by it matched the domain of the target ontology accurately

- A score of 2 points was assigned to a method if the result produced by it matched the sub or super categories of the target ontology domain.
- A score of 1 point was assigned to a method if the result produced by it was somewhat related to the domain of the target ontology.
- No score was assigned for a misclassification.

The scoring was done in two different ways:

- Cumulative score for each method was calculated inorder to find the performance of each method.
- Cumulative score was for each method was calculated by grouping ontologies of similar kind (superficial, elaborate and medium).

5.5.1 CUMULATIVE SCORES FOR EACH METHOD

The following table summarizes cumulative scores for each method

Table 5.3 Cumulative scores for each method

OntoClass with Target Ontology Classes	OntoClass with Target Ontology Text	OntoClass with Wikipedia Miner Toolkit
0.88	0.81	0.66

5.5.2 CUMULATIVE SCORES BASED ON TYPE OF ONTOLOGIES

As per the evaluation done in the previous sections, the target ontologies were evaluated as superficial, medium and elaborate. Accordingly, there are four superficial ontologies, four medium ontologies and twelve elaborate ontologies. A cumulative score for each group of target ontolgies was calculated for every method. The following table summarizes the scores:

Table 5.4 Cumulative scores for three groups of ontologies

Type of Ontology	OntoClass with Target Ontology Classes	OntoClass with Target Ontology Classes and Text	OntoClass with Wikipedia Miner Toolkit
Superficial	0.91	0.83	0.58
Medium	0.83	0.83	0.50
Elaborate	0.88	0.80	0.70

5.6 OVERALL PERFORMANCE

The results show that the first method is the most efficient in categorizing a given target ontology. The second method uses more text from the target ontology but did not perform well as it skews the classification of the categorizer. The categorization results are the best with elaborate ontologies followed by superficial and then medium ontologies. The third method did not perform as good as the first two methods. It gave a more broad classification in most of the cases due to absence of a relationship graph for the ontology classes.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Ontologies are one of the major sources of knowledge and data. With the emergence of Semantic Web, there is an increasing interest in developing and finding new ontologies describing different domains. With so much research in the Semantic Web area, several ontologies have come into existence. Today, there exist many ontologies even to describe a particular domain of knowledge. With so many ontologies describing the same domain of knowledge, there are many tools which provide mappings between them to show how similar they are. Ontology alignment or Ontology mapping provides interconnections between different ontologies representing the same domain of knowledge.

Apart from ontology alignment, searching for newly developed ontologies is a difficult task. There are search engines like Swoogle which can be used to search for a particular ontology, but it uses only keywords to provide the required results. It does not perform a semantic search for ontologies which might result in some ontologies being left out of the search results since they are not being recognized semantically as well.

The system we have built is a Ontology Categorization tool called OntoClass which takes an unknown target ontology and finds how well it fits into a given reference ontology and then categorizes it. This system uses Wikipedia as the reference ontology. Wikipedia is a rich source of information being used in natural language processing and also in a variety of research areas dealing with the Semantic Web. It represents a giant database of concepts and semantic

relationships. We made a use of ontology from Wikipedia because it covers almost all domains. This was done by three methods. First method used target ontology class names and a semantic graph module to generate important nodes based on centrality values. Node scores were calculated based on these values. Target ontology class names were compared with semantic graph nodes to find a match. For no match found, synonyms of semantic graph nodes were compared with the target ontology class names and a list of final matches was created. This list along with the important nodes list was sent to the categorizer to get the final categories. Second method used target ontology text (i.e. class names, labels, comments and properties) in the first step along with semantic graph to generate important nodes based on centrality values. Node scores were calculated based on these values. Target ontology class names were compared with semantic graph nodes to find a match. For no match found, synonyms of semantic graph nodes were compared with the target ontology class names and a list of final matches was created. This list along with the important nodes list was sent to the categorizer to get the final categories. Third method uses a toolkit called Wikipedia Miner to navigate and use Wikipedia's content and structure. For each target ontology class, corresponding mapped entities were found from Wikipedia Miner. For each mapped Wikipedia entity category, levels of parent categories were found till a parent category that covers almost all the entities was discovered. The dominant category was reported as the category of the target ontology. This system would be very useful to categorize unknown ontologies.

6.2 FUTURE WORK

The methods used to classify a given target ontology do not consider annotations and instances while parsing the given target ontology. These methods could be further extended to do so. The third method could be improved by obtaining a graph from Wikipedia miner matches to

indicate the most important nodes. The existing system can be combined with a Semantic search engine such as Swoogle to increase the efficiency of search results. By using multiple target ontologies against the reference ontology, this system can be further extended to compare given two target ontologies. The classification results can be used as a metric for evaluating the quality of a given ontology.

CHAPTER 7

REFERENCES

1. Berners-Lee, Tim., & Fischetti, Mark. (1999) Weaving the Web. Harper, SanFrancisco.

2. Web: http://en.wikibooks.org/wiki/Semantic_Web

3. W3C. Web: http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#%2824%29.

4. T. R. Gruber, T.R. (1993). A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220.

5. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., & Kolari, P. (2005). Finding and Ranking Knowledge on the Semantic Web. In Y. Gil, E. Motta, V. Benjamins & M. Musen (Eds.), The Semantic Web – ISWC 2005 (Vol.3729, pp. 156-170): Springer Berlin / Heidelberg.

6. W3C. Web: http://www.w3.org/RDF/

7. Neon-toolkit- XDTools. Web: http://neon-toolkit.org/wiki/2.3.1/XDTools

8. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., et al. (2009). DBpedia - A crystallization point for the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web, 7(3), 154 - 165.

9. W3C. Web: http://www.w3.org/TR/owl-ref/

10. W3C. Web: http://www.w3.org/TR/owl-guide/

11. Ehrig, M. (2007). Ontology Alignment: Bridging the Semantic Gap.

12. Spiliopoulos, V., Valarakos, A. G., & Vouros, G. A. (2008). CSR: discovering subsumption relations for the alignment of ontologies. Paper presented at the Proceedings of the 5th European semantic web conference on The semantic web: research and applications.

13. Jérôme Euzenat., & Svaiko,P. (2007). Ontology Matching: Springer-Verlag.

14. Wikipedia Statistics. Web: http://stats.wikimedia.org/EN/TablesArticlesTotal.htm

15. Milne, D., & Witten, I.H. (2009) An Open-Source Toolkit for Mining Wikipedia.

16. Wikipedia Miner. Web: http://wikipedia-miner.sourceforge.net/

17. The Java Universal Network/Graph Framework or JUNG. Web:
http://jung.sourceforge.net/index.html

18. Janik, M., & Kochut, K. J. (2008). Training-less Ontology-based Text Categorization.

19. Janik, M., & Kochut, K. J. (2008). Wikipedia in Action: Ontological Knowledge in Text Categorization. Paper presented at the Proceedings of the 2008 IEEE International Conference on Semantic Computing.

20. Fellbaum, C., & NetLibrary Inc. (1999). WordNet an electronic lexical database, Language, speech, and communication.

21. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., et al. (2004). Swoogle: a search and metadata engine for the Semantic Web. Paper presented at the Proceedings of the thirteenth ACM international conference on Information and knowledge management.

22. Samir, T., Arpinar, I. B., Michael, M., Amit, P. S., & Boanerges, A.-M. (2005). {OntoQA}: Metric-Based Ontology Quality Analysis, Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources.

23. Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating Semantic Web contents with Protege-2000. *Intelligent Systems, IEEE, 16*(2), 60-71.