## ABSTRACT

## ONTOLOGY LEARNING FOR THE SEMANTIC DEEP WEB

**by**
**Yoo Jung An**

Ontologies could play an important role in assisting users in their search for Web pages. This dissertation considers the problem of constructing natural ontologies that support users in their Web search efforts and increase the number of relevant Web pages that are returned. To achieve this goal, this thesis suggests combining the Deep Web information, which consists of dynamically generated Web pages and cannot be indexed by the existing automated Web crawlers, with ontologies, resulting in the Semantic Deep Web. The Deep Web information is exploited in three different ways: extracting attributes from the Deep Web data sources automatically, generating domain ontologies from the Deep Web automatically, and extracting instances from the Deep Web to enhance the domain ontologies. Several algorithms for the above mentioned tasks are presented. Experimental results suggest that the proposed methods assist users with finding more relevant Web sites. Another contribution of this dissertation includes developing a methodology to evaluate existing general purpose ontologies using the Web as a corpus. The *quality of ontologies* (QoO) is quantified by analyzing existing ontologies to get numeric measures of how natural their concepts and their relationships are. This methodology was first applied to several major, popular ontologies, such as WordNet, OpenCyc and the UMLS. Subsequently the domain ontologies developed in this research were evaluated from the naturalness perspective.

# ONTOLOGY LEARNING FOR THE SEMANTIC DEEP WEB

by
Yoo Jung An

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2008

**APPROVAL PAGE**

**ONTOLOGY LEARNING FOR THE SEMANTIC DEEP WEB**

**Yoo Jung An**

Dr. James Geller, Dissertation Advisor — Date
Professor, Department of Computer Science, NJIT

Dr. Narain Gehani, Committee Member — Date
Professor, Department of Computer Science, NJIT
Dean, College of Computing Sciences, NJIT

Dr. Dimitrios Theodoratos, Committee Member — Date
Associate Professor, Department of Computer Science, NJIT

Dr. Brook Yi-fang Wu, Committee Member — Date
Associate Professor, Department of Information Systems, NJIT

Dr. Yugyung Lee, Committee Member — Date
Associate Professor, Department of Computer Science Electrical Engineering,
University of Missouri at Kansas City

## BIOGRAPHICAL SKETCH

**Author:** Yoo Jung An

**Degree:** Doctor of Philosophy

**Date:** January 2008

**Undergraduate and Graduate Education:**

- Doctor of Philosophy in Computer Science
  New Jersey Institute of Technology, Newark, NJ, January 2008

- Master of Science in Computer Science
  New Jersey Institute of Technology, Newark, NJ, January 2004

- Master of Science in Consumer Science
  Dongguk University, Seoul, Korea, August 1995

- Bachelor of Science in Home Economics Education
  Kookmin University, Seoul, Korea, February 1993

**Presentations and Publications:**

**Journal Articles:**

Yoo Jung An, Kuo-Chuan Huang and James Geller,
"Naturalness of Ontology Concepts for Rating Aspects of the Semantic Web,"
Communications of the International Information Management Association,
vol. 6, no. 3, pp. 63-76, 2006.

Yoo Jung An, Kuo-Chuan Huang, and James Geller
"Comparative Anatomy of Ontologies: the Semantic Naturalness Perspective,"
Second Review for Publication.

**Conference Papers:**

Soon Ae Chun, Yoo Jung An, James Geller, and Sunju Park,
"Fuzzy Virtual Card Agent for Customizing Divisible Card Payments,"
Proceedings of E–Commerce and Web Technologies, 6th International Conference, EC–Web 2005, Copenhagen, Denmark, 2005, 287-296.

Yoo Jung An, James Geller, Yi-Ta Wu and Soon Ae Chun,
"Semantic Deep Web: Automatic Attribute Extraction from the Deep Web Data Sources,"
Proceedings of the 22nd Annual ACM Symposium on Applied Computing, SAC 2007, Seoul, Korea, 2007, 1667-1672.

Yoo Jung An, James Geller, Yi-Ta Wu and Soon Ae Chun,
"Automatic Generation of Ontology from the Deep Web,"
Proceedings of the 6th International Workshop on Web Semantics, WebS07, Regensburg, Germany, 2007, 470-474.

**Posters:**

Yoo Jung An, Kuo-Chuan Huang, and James Geller,
"Rating the Naturalness of Ontology Taxonomies,"
Proceedings of the 20th International FLAIRS Conference, Key West, Florida, 2007, 176-177.

## ACKNOWLEDGMENT

My sincere and foremost thanks go to Professor James Geller. His continuous guidance and support allowed me to complete this dissertation successfully.

I am also grateful to the committee members, Dr. Narain Gehani, Dr. Dimitrios Theodoratos, Dr. Brook Yi-fang Wu and Dr. Yugyung Lee for reviewing the dissertation thoroughly and invaluable suggestions.

I extend my thanks to my fellow graduate students and colleagues at the Semantic Web and Ontologies lab. I benefited a lot from their invaluable discussions and encouragement.

Many parts of this dissertation have been published or submitted for publication. I extend my thanks to a co-author Dr. Soon Ae Chun for an excellent collaboration. I will also give a special thanks to Dr. Yi-Ta Wu and Kuo-Chuan Huang who have worked on the implementation.

Thanks are also due to Professor David Nassimi, Professor Andrew Sohn and Dean of College of Computing Sciences, Professor Narain Gehani for the teaching assistantship throughout my graduate years.

Finally, I am deeply grateful to my husband Dr. Sung-Hyuk Cha and my parents Jung Pyung An and Kum Hong Choi and my family for support and love.

## TABLE OF CONTENTS

**TABLE OF CONTENTS**
**(Continued)**

TABLE OF CONTENTS
(Continued)

## LIST OF TABLES

## LIST OF TABLES
## (Continued)

## LIST OF FIGURES

## LIST OF FIGURES
## (Continued)

## LIST OF FIGURES
## (Continued)

**Figure** **Page**

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

Different users use different search terms according to their knowledge and intuition to find relevant Web pages that they are looking for. The numbers of relevant Web pages returned to users differ dramatically, mainly depending on the terms entered into conventionally available Web search engines. Many Web pages returned to users may be completely irrelevant, and it takes too long for users to identify the relevant Web pages by going through too many results. It is necessary to develop a methodology such that the number of returned Web pages becomes smaller while the overall number of relevant Web pages becomes bigger.

Quoting the famous scientist (Hawking 2002), Galileo Galilei, "where the senses fail us, reason must step in." This quote can be interpreted as follows in the Web search problem. Different keywords or terms entered by users can be interpreted as "senses." Sometimes, the returned Web pages do not satisfy the users, i.e., there are too many returned Web pages many of which are irrelevant. These cases correspond to "where the senses fail us." Thus some logical methodology must step in to find more relevant Web pages. The methodologies of this dissertation are based on utilizing *ontologies*.

*Ontologies* can provide semantics to the next generation of the World-Wide Web, and thus, are considered as an important topic of Web research. Recently, the *Semantic Web* (Berners-Lee *et al.* 2001) has become a major research issue in several sub-disciplines of Computer Science and Information Systems. The goal of the Semantic

Web is to automate many tasks that humans perform with the World-Wide Web today. The vision of the Semantic Web relies on agent programs like *softbots* roaming the Web, finding data and services, combining them and returning them to their user. These agents will need some human-like knowledge to perform their tasks. For this purpose, *ontologies* will be sprinkled all over the Semantic Web.

It is imperative to build ontologies to achieve this goal, yet, very few domain ontologies are currently available and most are too small to be good resources. What is worse, existing ontologies often provide unnatural concept labels (Lee and Geller 2005).

Ontologies have often been constructed in one of the following two ways. Either, they are manually developed by experts in the domain. This approach to building domain ontologies has two disadvantages: 1) It is extremely time consuming to construct ontologies by hand, and 2) domain ontologies are generally based on the view point of the expert, and are subjective, and limited to some degree by the available expertise. The other approach to building ontolgies is using text mining techniques on numerous Web pages. In this approach, extracted concepts are prone to errors and relationships among concepts are hard to be extracted.

## 1.2 Approach

This thesis presents a method for the automatic extraction of ontologies from sources beyond unstructured text Web page sources. A large number of Web sites (i.e., Web data sources) provide a considerable amount of information with the support of backend databases and dynamic query schemes. The contents of the databases are dynamically retrieved by programs and accessible and searchable by manual query interfaces. At the

present time, they are not included in the results returned by search engines or indexed by Web crawlers. Thus, there exists a large amount of Web information that is not "visible" to general search engines. This "hidden" information in Web sites has been called the *Deep Web* (Singh 2002). Sometimes, the Deep Web is called the "hidden" or "invisible" Web. Accessing and utilizing the hidden Deep Web information is of interest and a contribution of this dissertation.

This dissertation considers the problem of searching Web pages faster and retrieving better results than currently available Web search engines by combining Semantic Web techniques and information from the Deep Web. Historically, research on the *Semantic Web* and on the *Deep Web* has happened in parallel, with little interaction between the two fields. This dissertation introduces the Semantic Deep Web, focusing on the semantics of information in Deep Web sources.

Automatic generation of domain ontologies from a variety of sources (e.g., existing ontologies, Deep Web data sources, etc.) is a challenging research topic that is attacked in this dissertation. This dissertation demonstrates three different aspects of utilizing Deep Web sources. The first aspect is to extract Web page attributes of Deep Web front ends automatically (An *et al.* 2007a). The second aspect is to extract ontologies from the Deep Web automatically (An *et al.* 2007b). The last aspect is to extract instances from the Deep Web automatically to enrich the ontologies by extending them downwards.

Furthermore, an evaluation methodology for existing ontologies has been developed in this research. It was tested with several major, popular ontologies, namely OpenCyc, WordNet and the UMLS. Then it was applied to the automatically generated

domain ontologies. Thus, this dissertation presents research on the creation on of a "Semantic Deep Web," identification of Deep Web sources, automatic generation of domain ontologies from the Web sources (An *et al.* 2007c), automatic instance extraction from the Deep Web and a novel methodology for the evaluation of existing ontologies (An *et al.* 2006, An *et al.* 2007b).

### 1.3 Overall Research Framework

Ontologies could provide a useful semantic layer for the Deep Web. On the other hand, the contents of domain ontologies can be enriched by Deep Web sources. Thus, the Deep Web and ontologies can mutually benefit each other, in the view of this dissertation. Hence, this thesis introduces a framework of the Web as the Semantic Deep Web (An *et al.* 2007a). Before discussing all the aspects for the Semantic Deep Web in depth, the main components of the Semantic Deep Web are given in Figure 1.1. Focusing on ontologies, two main aspects, the Deep Web and ontologies extend the WWW to the Semantic Deep Web.

The *Semantic Deep Web*, which is derived from a combination of aspects of the Deep Web and the Semantic Web, focuses on the search interfaces of the Deep Web and the ontology approach of the Semantic Web. An ontology provides semantic support by using controlled terms for concepts in a certain domain.

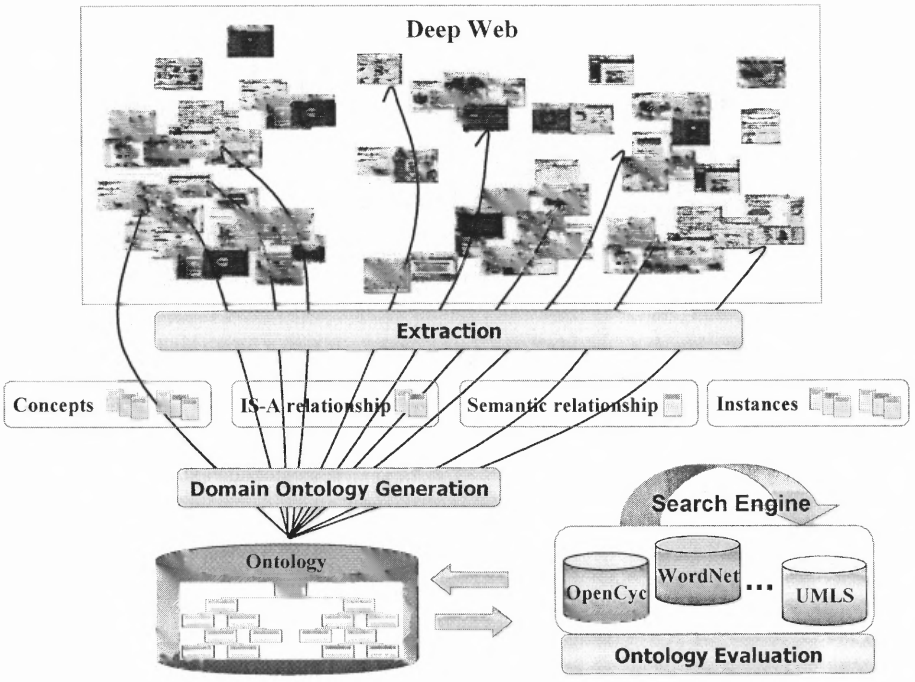



**Figure 1.1** The main components of the Semantic Deep Web.

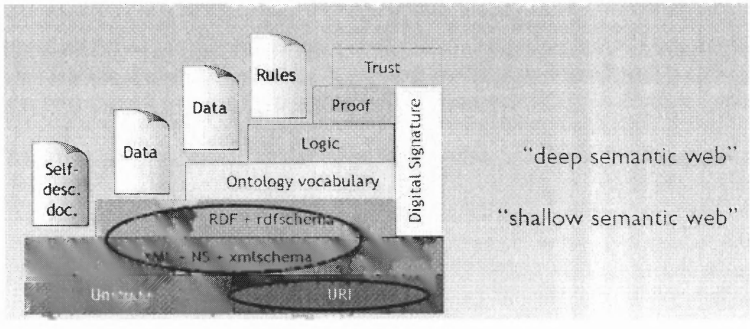



**Figure 1.2** Semantic Web layer cake (Bjelogrlic *et al.* 2006).

It should be noted that the Semantic Deep Web is not the same thing as the Deep Semantic Web. Researchers (Bjelogrlic et al. 2006) have distinguished between the Shallow Semantic Web and the Deep Semantic Web, referring to parts of the Semantic Web layer cake as shown in Figure 1.2. Hereby, the upper layers that are more Artificial

Intelligence oriented have been referred to as Deep Semantic Web. Here, on the other hand, a semantic layer is added to the Deep Web which was newly introduced in this research, independent from the Semantic Web. Thus, one could take the Semantic Deep Web and classify it into a Shallow Semantic Deep Web (SSDW) and a Deep Semantic Deep Web (DSDW). This dissertation considers only the Shallow Semantic Deep Web. The *Semantic Deep Web* was first presented in previous work (An *et al.* 2007a).

As the main components of the Semantic Deep Web, this dissertation research investigates **1)** Attributes of the Deep Web data sources and how to automatically extract them; **2)** Domain ontologies in *E-commerce* and how to automatically generate them from the Deep Web. **3)** Deep Web search and how to automatically extract instances from the Deep Web and include them in the domain ontologies and finally **4)** A methodology to evaluate the naturalness of existing general purpose ontologies, as well as the naturalness of the desired E-commerce ontologies.

### 1.4 The Process for Constructing the Semantic Deep Web

The proposed approach to constructing the Semantic Deep Web is depicted in Figure 1.3. In steps 1 (An *et al.* 2007a) and 2 (An *et al.* 2007c), attributes are automatically extracted from query interfaces of several domain-specific Deep Web sources, and domain ontologies are automatically generated based on attributes of the query pages and WordNet. In order to enhance a domain ontology, instances and new concepts based on results obtained by querying suitable Deep Web sites are added to the domain ontology in steps 3, 4, 5 and 6. As a new Web site is visited, new instances and concepts can be recognized and merged into the ontology. In step 7, the question is addressed how to

access Deep Web data from a site which is not designed to cooperate with existing crawling algorithms. In step 8, a novel approach is presented to evaluate the quality of several ontologies (WordNet, UMLS, etc.) to get numeric measures of what may be called "naturalness" (An *et al.* 2006; An *et al.* 2007b); the "naturalness" measurement was applied to the enriched domain ontologies as well.

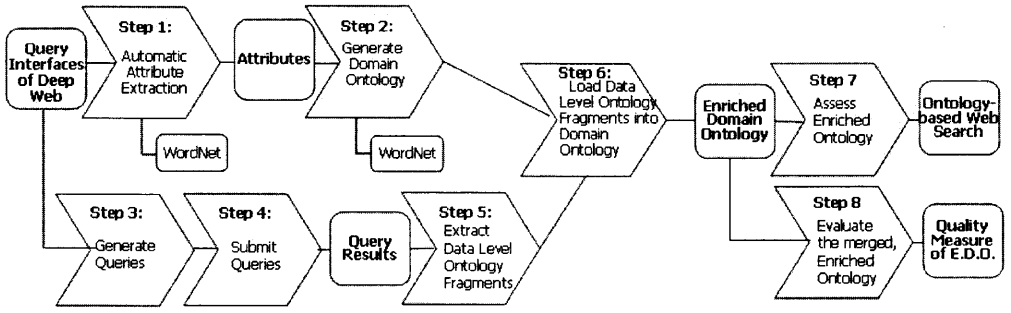



**Figure 1.3** Approach to building the Semantic Deep Web.

## 1.5 Organization

The rest of this dissertation is organized based on the steps described in Figure 1.2. Thereby, the outline of this dissertation is described as follows. Chapter 2 presents a method to extract attributes from the Deep Web automatically, i.e., step 1 in Figure 1.3, which was published in (An *et al.* 2007a). Chapter 3 introduces a method to generate ontologies automatically from the Deep Web, i.e., step 2 in Figure 1.3, which was published in (An *et al.* 2007c). Chapter 4 covers the steps 3, 4, 5, 6, and 7 which extract instances from the Deep Web automatically to enrich the ontology by extending them downwards. Results of an experimental assessment of the enriched ontology will also be reported to check whether the proposed method increases the number of relevant result Web pages. Chapter 5 presents a statistical evaluation of the naturalness measurements. A

summary and conclusions are in Chapter 6. Finally, future work and limitations are discussed in Chapter 7.

# CHAPTER 2

# AUTOMATIC ATTRIBUTE EXTRACTION FROM THE DEEP WEB

## 2.1 Introduction

This chapter introduces automatic attribute extraction from the Deep Web, utilizing an ontology to determine attributes to access the Deep Web.[1] The *Automatic Attribute Extraction* method (1) identifies attributes that are used by query Web page designers, called Programmer Viewpoint Attributes, and (2) attributes that are presented as labels to users, called User Viewpoint Attributes. An ontology enriches the candidate query attributes by providing synonyms and by matching attributes used by designers and users.

Dynamic query interfaces with HTML <form> elements have been the major method for accessing the Deep Web. In (Ntoulas *et al.* 2005), information in Deep Web sites was categorized as being represented either in textual or structured databases. While a textual database needs a single input keyword for searching text documents, a structured database typically requires a user to fill in several input fields of a query interface, so that the system will produce and return a search result. In order to display a set of data, complex forms need to be serialized, to ensure the syntactic correctness of a user's request before extracting data from the backend database. This dissertation work focuses on such structured databases.

The recent interest in the Deep Web has resulted in a new research focus on information hidden inside Web-accessible databases. For example, Ipeirotis (2004) has

[1] This chapter's contents were published in (An *et al.* 2007a).

studied information hidden behind query interfaces, focusing on *hidden-Web text database*.

Retrieval of information from the Deep Web is often desired. For example, a Web user may be looking for a cheap airline ticket. For this purpose she needs to scroll through prices after entering necessary values (e.g., departure date and time) into a query form on a website. Information about airline tickets she or he is viewing is "structured (Arasu and Garcia-Molina 2003)" such that she or he can see the results organized by different criteria. Some users prefer cheap flights, while other users prefer direct flights or want to fly only with certain airlines. To locate an appropriate website, most users would enter a few keywords ("cheap tickets") into a general purpose search engine such as Google. Typically such a search will locate the major existing airline reservation websites.

However, a user may still be unsatisfied with the query results. The reason is that the websites found by a general purpose search engine are usually the most popular sites. This does not guarantee that these sites will have the lowest ticket prices. A new startup may offer better prices, but it will be almost impossible to locate this site, because it will be hidden among the thousands of results of a general search engine. The user wants to find the Web data sources, even non-popular ones, which are most relevant to her needs. This is one example of the "information extraction problem (Arasu and Garcia-Molina 2003)."

In order to support this kind of Web search requirement, researchers involved in Web technology have studied several issues regarding the Deep Web. These include how to understand the context of queries and give proper responses (Singh 2002), how to best

processes queries to return relevant pages (Singh 2002), how to create dynamic pages that need computation on the fly, how to support simultaneous searches (Ghanem and Aref 2004), and how to classify the content of Web databases (Ipeirotis 2004). These approaches reflect different characteristics of the Deep Web.

Understanding the attributes and contents of Deep Web data sources is also important in order to locate the most relevant Deep Web data sources for a user, since these sources use different attributes to access contents. Kabra *et al.* (2005) presented an attribute co-occurrence framework to rank and select Deep Web data sources based on the users' requirements. They first construct a co-occurrence graph to indicate the relationships between pairs of attributes. Then, an attribute relevance score is calculated, indicating how likely it is that a query interface with this attribute will be of interest to the user. Finally, the score for each data source is obtained based on all the attribute relevance scores. Although Kabra *et al.*'s algorithm successfully allows the user to input an imprecise initial query, the attributes of their data sources were obtained manually. The reason why they manually determined the attributes of each used website is that automatic attribute extraction has been proven to be a difficult task (Raghavan and Garcia-Molina 2001).

The newly proposed approach in this dissertation is to extract attributes automatically by using the viewpoint of the user and the viewpoint of the (Web application) programmer and reconciling the results obtained in this way with the help of an ontology. Ontologies, together with agent technologies, are primary ingredients of what Berners-Lee *et al.* (2001) have called the Semantic Web. Historically, research on the Semantic Web and on the Deep Web has happened in parallel, with little interaction

between the two fields. The Semantic Deep Web, combining the Semantic Web and the Deep Web is introduced in this dissertation work.

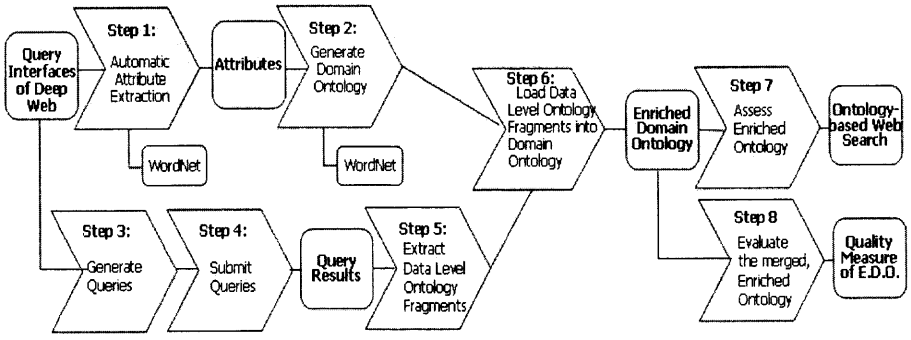



**Figure 2.1** The frame work for extracting attributes of Deep Web sources.

Chapter 2 is about a part of the overall system architecture in Figure 2.1 and marked as Step 1. The rest of this chapter is organized as follows. In Section 2.2, definitions of terms used in this chapter and the rest of the dissertation are given. The Section 2.3 presents the approach to the automatic attribute extraction from the Deep Web data sources, including *Programmer Viewpoint Attribute* (PVA), *User Viewpoint Attribute* (UVA), and final attribute extraction algorithms. Finally, Section 2.4 presents the implementation of the algorithms and the effectiveness of final attributes for user queries.

## 2.2 Definitions

In this section, the terms that appear in this research are delineated. Before describing the newly developed approach to automatically extracting the proper attributes from Web data sources of the Deep Web, the meaning of the ambiguous term "*attribute*" is first clarified. In the general sense, an attribute of a Web data source is any item of

information that describes this source. The more specific meaning of "attribute" is derived from the HTML/XML syntax. A tag of HTML consists of a mandatory name between angular brackets, which may be followed by optional attribute/value pairs. As an example, the Web page in Figure 2.2 is generated by the HTML code in Figure 2.3, which contains several attribute/value pairs. Thus the tag <SELECT> contains the attribute "size" with the value 2. The attribute and the value of such a pair are separated by an equal sign (=).

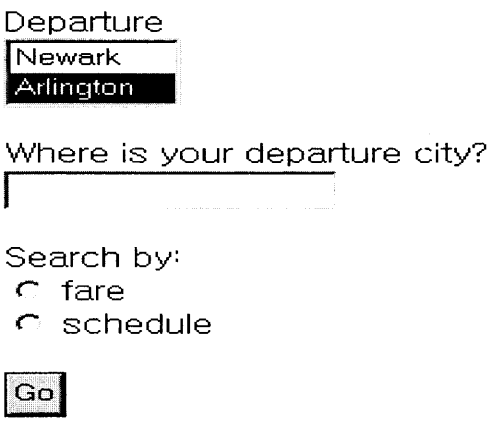



**Figure 2.2** A simple Form Example.

In order to reduce the confusion that might be caused by this ambiguity, the two meanings of "attribute" based on two possible viewpoints are distinguished. A user sees the text of a Web data source and the text areas into which she is supposed to enter information. Usually a text string is close to (above or next to) each text area, indicating what kind of information needs to be filled into this text area. A user normally does not see the attribute/value pairs of the HTML code, although she could, of course, view the HTML source of the page. The latter is typically not done, unless it is needed.

On the other hand, a Web application programmer will primarily need to look at the attribute/value pairs used within HTML forms, as those pairs define the only way for accessing the form information from within the application. While the programmer might occasionally look at the form itself, he will try to compress everything he needs to know about the form into the attribute/value labels of the form. Thus, he has every interest in making those labels meaningful. Following this distinction, Programmer Viewpoint Attributes (PVAs) are distinguished from User Viewpoint Attributes (UVAs). PVAs are extracted from within HTML tags whereas UVAs are the results of analyzing the text of the Web form, especially as it is associated with text entry areas. It should be noted that in (World Wide Web Consortium 1999) elements on a form are called "controls" which are used as a medium for user interaction.

In the process of locating Deep Web sources, many attributes will be extracted, some of which might not be used in the final analysis. Such attributes are referred to candidate attributes. Both PVAs and UVAs are utilized to achieve the following three goals: (1) to semantically categorize the encountered Deep Web data sources, (2) to recommend Deep Web sources which are likely to be of interest to the user, and (3) to solve this limited form of the "information extraction problem." By comparing PVAs and UVAs and using an ontology to recognize synonyms, it becomes easier for the program to ascertain the meaning of the entry fields on a Deep Web data source. The final results of the attribute analysis involving both PVAs and UVAs are referred to as Final Attributes, or in short, FAs.

The simple <FORM> in Figure 2.3 shows the HTML code of a number of common form elements. The <LABEL>, <SELECT>, <OPTION> and <INPUT> elements all contain PVAs.

```
<FORM action= "..." method= "...">
 <P><LABEL for= "departure_city">Departure<BR>
    </LABEL>
 <SELECT size= "2" name= "depart_city">
    <OPTION selected value= "city1">Newark</OPTION>
    <OPTION>Arlington</OPTION></SELECT><P>
   Where is your departure city? <BR>
<INPUT type= "text" id= "origin"><P> Search by:<BR>
 <INPUT type= "radio" name="searchBy" value="fare"> fare<BR>
 <INPUT type="radio" name="searchBy" value= "schedule"> schedule<P>
  <INPUT type= "submit" value= "Go"> </P> </FORM>
```

**Figure 2.3** HTML corresponding to Figure 2.2.

Comparing Figure 2.2 and Figure 2.3 again, it becomes immediately clear that both viewpoints indicate that this form might be related to an airline reservation system, but in slightly different ways. Figure 2.2 mentions "Departure City" according to the user viewpoint. Figure 2.3 mentions departure_city according to the programmer viewpoint.

During PVA extraction, the values of the id, name and value attribute/value pairs of HTML INPUT and SELECT elements are used to generate the PVAs. For example, "depart_city," "origin," "searchBy," and "schedule" are selected for PVA extraction. During UVA extraction, the English text is used to form a set of UVAs. For example, "Where is your departure city?" provides relevant information. Finally, by comparing PVAs and UVAs, if necessary by looking up synonyms, FAs can be determined. For example, "departure city" will become one of the FAs.

Following are the definitions used in this dissertation work.

**Definition 1 (Web Data Source)**: A Web Data Source is a set of Web pages $\{DS_1, DS_2, \cdots, DS_n\}$ written in HTML, with each $DS_i$ containing a set of HTML form elements $HF_i$. □

**Definition 2 (Inner Identifier)**: An inner identifier is the value of an attribute/value pair of a tag, where the attribute is "name," or"id". The attributes of some tags (e.g., <SUBMIT>, <RESET>, <HIDDEN> or <IMAGE> ) are excluded. □

$II_i^k$ is used to denote the set of $k$ inner identifiers of one Web data source.

Example: For <SELECT size=2 name="depart_city">, the inner identifier is "depart_city."

**Definition 3 (Keywords)**: A label L consists a set of keywords, KL, that represent all the domain-specific content words that are found between the begin tag <LABEL> and/or <SELECT> and the end tag </LABEL> and/or </SELECT>. The set of key words of a Web Data source, denoted as KW ={$KL_1$, $KL_2$, … $KL_k$}where $k$ is the number of keywords is defined as the union of all keywords of all labels in a Web Data Source.□

Example: For <LABEL for="departure_city">Departure</LABEL> the set of keywords consists of {Departure}. For <OPTION value="0000" selected>Any Time</OPTION> the set of keywords consists of {Any, Time}.

KW is used to break inner identifiers that may consist of several words into these constituting words. For example, if KW={seat, infant} and an inner identifier is given as "abcinfantwithseat2" this will be broken down into a sequence abc, infant, with, seat, 2.

An inner identifier may consist of several words, combined with a separator or just concatenated, e.g., departure_month, departureMonth, or departuremonth. In order

to increase the likelihood of matches, the component words of an inner identifier are allowed to be attributes themselves or as pairs or triples (n-tuples) in their original order. Thus, for departure_month, the following three are considered possible attributes: "month," "departure," "departure month." These possible attributes are *candidate attributes* for this domain.

**Definition 4 (Inner Identifier-based Candidate Attribute)**: An Inner Identifier-based Candidate Attribute, IICA, is a candidate attribute which has been derived from an inner identifier. □

Based on the bag of all IICAs of all Web Data Sources, the Programmer View Attributes, or PVAs in short are defined as follows.

**Definition 5 (Programmer Viewpoint Attribute)**: A Programmer View Point Attribute, PVA, is an Inner Identifier-based Candidate Attribute that occurs more than once in the bag of all IICAs. □

In order to eliminate spelling errors and apparently unimportant attributes, candidate attributes that occur only once over all sampled $HF_i$ are dropped. So as to increase the likelihood of matches, synonyms of PVAs are added to the set of all PVAs.

**Definition 6 (Synonym of PVA)**: A synonym of a programmer viewpoint attribute, SOPVA, is a (set of) words from an ontology that are synonyms of the PVA. □

WordNet is used as the ontology in this dissertation. The set of SOPVAs contains synonyms of all PVAs including the PVAs themselves in a singular format.

Up to now, attributes that have been extracted are based only on information within starting tags. Now let's turn to attributes based on the free text between pairs of HTML tags. For this purpose, <OPTION> tags and <LABEL> tags shall be ignored. Most such free text will be derived from locations close to <INPUT> tags, as those usually indicate what values are expected in an input field.

**Definition 7 (Free Text-based Candidate Attribute)**: A free text-based candidate attribute FTCA is a sequence of English words and symbols (such as   or /) found between a pair of tags in close proximity to an <INPUT> tag. □

$FTCA_i$ shall be used for the set of free text based candidate attributes of one Web data source. The fuzzy term "close proximity" shall be employed because HTML often contains intervening tags such as <BR> which are meaningless separators.

**Definition 8 (User Viewpoint Attribute)**: A User Viewpoint Attribute, denoted as UVA, is a sequence of English words derived from an FTCA by eliminating special symbols. □

$UVA_i$ represents the set of user viewpoint attributes of one Web data source. Different special symbols require different forms of processing. Most special symbols can be replaced by blanks. However, when a slash occurs, then the free text is broken into two separate User Viewpoint Attributes.

**Definition 9 (Synonym of UVA)**: A synonym of a user viewpoint attribute SOUVA is a (set of) words from an ontology that are synonyms of the UVA. □

$SOUVA_i$ represents the set of synonyms of user viewpoint attributes of one Web data source where the UVAs themselves are included in a singular format.

**Definition 10 (Final Attribute)**: A final attribute, **FA,** is an element of $SOUVA_i$ which has a sub-string (or a sub-string of one of its synonyms) matches one element of *SOPVA* with larger than $\alpha$ % block occupancy rate of the elements among all consecutive characters of the UVA, where $0 < \alpha \leq 100$. $\alpha$ is a factor that may affect the total number of final attributes. That is, the smaller $\alpha$ is, the larger the number of final attributes. □

Example: it can be determined whether a target UVA "departure date" can be considered as a final attribute or not where $\alpha$ =50%. Suppose there exists a target PVA "departure" in SOPVA. Then, since the sub-string "departure" in the target UVA is larger than 50% (9/14) of the target PVA's length, the target UVA "departure date" will be considered as one of the final attributes of the Web data source.

$FA_i$ represents the set of final attributes of one Web data source and is used to describe the Web source to a user or a search engine. The union final attribute, UFA, is used to denote the union of all sets of final attributes. The detailed procedures for deriving the FAs will be presented in the following Sections.

## 2.3 Automatic Attribute Extraction (AAE) from Deep Web Data Sources

### 2.3.1 Approach to AAE

In this section, an overview of the *Automatic Attribute Extraction* (AAE) algorithm, which is based on the following three primary observations, is presented.

(1) The query interface between the clients (users) and the server provides a way for the users to submit their requirements.

(2) The context provided by the English language text in the query interface not only describes the services of the Website, but also assists the users with entering their requirements correctly.

(3) HTML tags often contain attribute/value pairs after the tag name. The values, called inner identifiers, within these HTML markup elements are developed by programmers and will presumably be named in a way that the programmers can easily remember the usages of data entry fields and the corresponding data to be expected.

The query interface has the purpose of processing the user's query actions. Therefore, it is likely that there exists a hidden database table containing the hidden attributes in a Web data source which correspond to fields in the query interface. The context provided by the English language text in the query interface is utilized to inform the users what information to enter into the form. Thus this text must describe the kind of data to be entered into the associated form field. These data items will then be used for query, insert, or update operations in a hidden backend database. There is a clear tradeoff between the usefulness of the attribute/value pairs within HTML elements and the free text between them. The free text must be in a format understandable to humans, otherwise humans could not use the interface at all. On the other hand, free text in a form often appears in phrases or even full sentences which are harder to process than single words. Values of attribute/value pairs typically come as single words. However, because the user cannot see these words, the interface builders often use inner identifier names which are not meaningful to users, such as FIELD3 or REDINFSO. Thus, inner identifiers have not been a major research issue in previous papers. However, they may provide important clues and are utilized in this dissertation work to contribute to a novel approach towards attribute determination.

Processing free text to derive UVAs is difficult for a machine due to at least the following two problems. First, it is difficult for a computer to distinguish the useful key words from the unnecessary words of a Web data source, since a word might be

important in certain applications, but useless in other areas. For example, when booking an airline ticket, "from" and "to" are important key words. However, in most other contexts, "from" and "to" would be considered stop-words and eliminated from consideration together with "the", "this," "an" and other stop-words. Secondly, it is difficult to determine the precise relationship between a key word, its corresponding hidden attribute and the corresponding column in the hidden database. In some interfaces, a key word might be a synonym for the corresponding attribute in the hidden table, but this would be a lucky case. In many cases only the programmer will know why he named a specific field in a specific way.

The newly developed approach to extracting the proper attributes for a Web source is shown in Figure 2.4. In Figure 2.4, there is an overlapping area between the UVAs and PVAs. That is, the final attributes will be determined by comparing UVAs and PVAs. In order to perform the goal of automatically determining the proper attributes for a Web data source, it is necessary to make use of all the information available in a flexible way. For example, the Web page designer might have called an input element "departureCity" in the HTML input markup element, <input name="departureCity" class ="smallform" type="text" size="20" value=""> indicating that the meaning of the input element is "departure city," which is close to the more useful "departure airport."

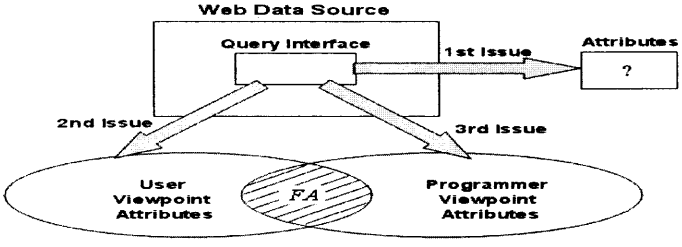



**Figure 2.4** Three important issues in attribute extraction.

Therefore, in order to automatically extract the attributes for each Web data source, a three-stage algorithm has been developed. Given a set of Web data sources, the PVAs are obtained from the inner identifiers of all the Web data sources. Secondly, the UVAs are obtained from the free text within the query interface. Lastly, the final attributes (FAs) of each Web data source are determined based on PVAs and UVAs by utilizing an ontology (WordNet). Note that, PVA and UVA extraction, and FA determination are all achieved automatically.

Figure 2.5 graphically displays the high level Automatic Attribute Extraction algorithm, the pseudo-code of which is shown in Algorithm 2.1. The details of each step will be described in Sections 2.4, 2.5 and 2.6. Looking at Figure 2.5, it becomes clear that the given algorithm does not treat PVAs and UVAs in a symmetrical way. Only one PVA set is matched against several UVA sets. This avoids the inefficiency of mapping $n$ PVA sets against $m$ UVA sets. It is the assumption that the PVA set is used as reference set, because it is likely to be more precise than the UVAs. The reason for that is that terms in the PVA are likely to be tied to terms from an underlying database schema.

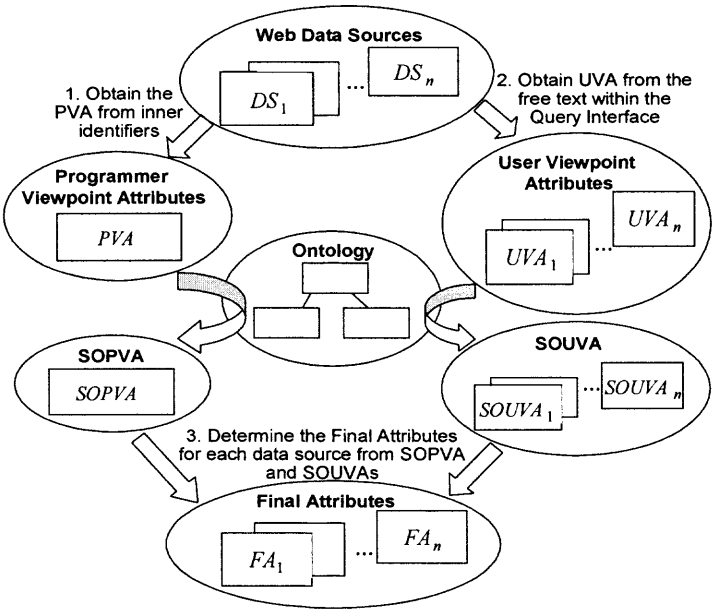



**Figure 2.5** Automatic attribute extraction components.

**Algorithm 2.1: Automatic Attribute Extraction**
**Procedure** AAE
**begin**
  **for** each $DS_i$ **do begin**
    Obtain $HF_i$ from $DS_i$
    Obtain $II_i^k$ from $HF_i$ (Alg. 2.2)
  **end**
  Obtain KW
  Obtain PVA and *SOPVA* (Alg. 2.3)
  **for** each $DS_i$ **do begin**
    Obtain $UVA_i$ and $SOUVA_i$
    Obtain $FA_i$ by comparing *SOPVA* and $SOUVA_i$
  **end**
**end**

### 2.3.2 PVA Extraction

While inner identifiers can be easily obtained from HTML elements by a program, they cannot be directly used for further analysis since they are usually comprised of several words and symbols. Therefore, the inner identifiers have to be further separated into several independent words. The algorithm 2.2 shows steps for separating a set of inner identifiers of a Web data source $DS_i$. The pre-condition for this algorithm is the existence of KW as defined in definition 3. The automatic derivation of KW is relatively simple by text extraction, thus detailed description is omitted.

**Algorithm 2.2: Separating the Set of Inner Identifiers**
**function** $SSII(II_i)$: $IICA_i$;
**begin** Remove the duplicated inner identifiers in set $II_i$.
  **for** each inner identifier in $II_i$ **do begin**
   **if** the inner identifier contains special symbols
     (:,/,{,},@,[,],>,$,&,#,+,\,.,=,?,;,*,_,{,",},<, etc.) **then**
     separate the inner identifier into several sub-strings;
   **if** each sub-string contains a Capital letter
        (i.e., camel case) **then**
      break each sub-string into several sub-strings;
   **for** each sub-string **do begin**
     **for** each key word of KW **do begin**
      **if** the key word is located in the sub-string **then**
          break each sub-string into several sub-strings with
            respect to the key word; **end end**
   obtain the separated inner identifier which is a string
        containing several sub-strings
  **end**
  **for** each separated inner identifier **do begin**
    count the number of sub-strings, *ss*, in the separated
        inner identifier
    **for** index *i* = 1 to *ss* **do begin**
      extract a string which is composed of *i*-word consecutive
        words from the separated inner identifier, and
        add the string into the set $IICA_i$; **end**
  **end**
   Remove the duplicated strings in $IICA_i$; **return** $IICA_i$
**end**

After obtaining the inner identifier based candidate attributes of each Web data source, the set of PVAs is obtained from all sets $IICA_i$ by the Algorithm 2.3.

**Algorithm 2.3: Obtaining PVA**
**function** OPVA(all of $II_i$): PVA;
**begin**
  **for** each $II_i$ **do begin**
    Obtain $IICA_i$ by calling function SSII($II_i$) (Algorithm 2.2)
  **end**
  **for** *IICA* in PVA **do begin**
    **if** *IICA* appears one time in the PVA **then**
      Remove the *IICA* from the set PVA
    **if** *IICA* contains several copies in PVA, **then**
      Keep one copy and Remove the duplicated ones
  **end**
  **return** PVA
**end**

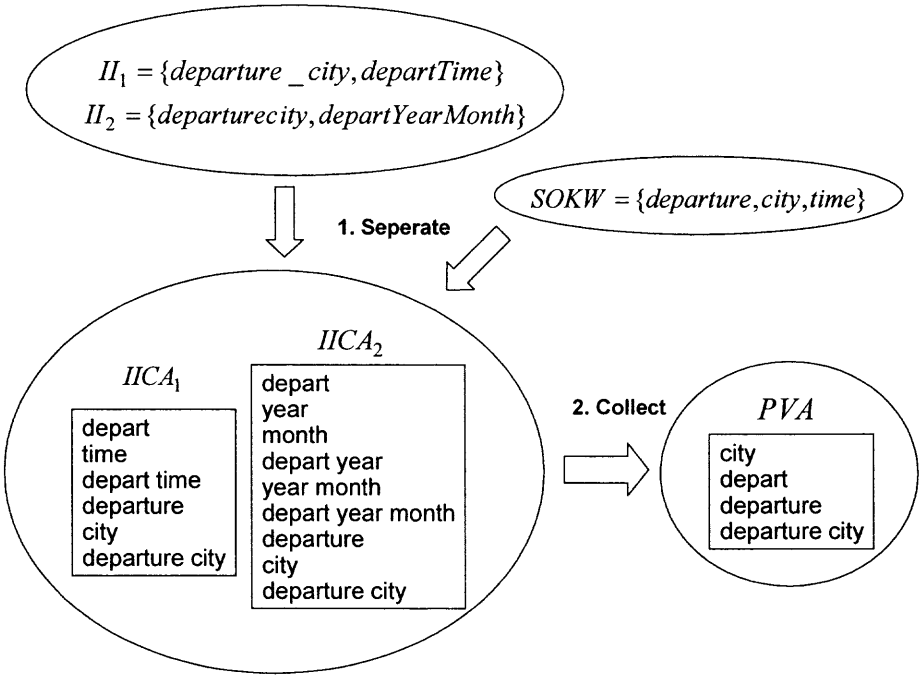



**Figure 2.6** An example of obtaining PVA.

Figure 2.6 shows an example of obtaining the Programmer Viewpoint Attributes. Suppose there are two Web data sources $DS_1$ and $DS_2$. First, two inner identifier sets, $II_1$ and $II_2$ are extracted from $HF_1$ and $HF_2$, respectively. $IICA_1$ and $IICA_2$ can be derived from $II_1$ and $II_2$ accordingly by splitting the inner identifiers using the Algorithm 2.2. Finally, the set PVA is obtained by the Algorithm 2.3.

### 2.3.3 UVA Extraction

The User Viewpoint Attributes are utilized to determine the final attributes of each Web data source, and they are obtained from the free text within the query interface. The procedure of obtaining the UVA for each Web data source, shown in Algorithm 2.4, requires that the free text between two HTML tags which potentially embodies semantics is added into the set $FTCA_i$. The text between <OPTION> and </OPTION> should be ignored since it does not describe attributes but instances.

**Algorithm 2.4: Obtaining UVA**
**function** OUVA($HF_i$ ): $UVA_i$;
**begin**
Remove all the text between <option> and </option> from $HF_i$ .
Obtain all free text between two HTML tags from $HF_i$ and add them as strings into set $FTCA_i$ .
**for** each string in set $FTCA_i$ **do begin**
**if** a string contains special symbols **then**
Separate the string into sub-strings with respect to the symbols, to obtain several free text based candidate attributes (*FTCAs*);
Add all *FTCAs* into set $UVA_i$
**end**
Remove the duplicated *FTCAs* in $UVA_i$
**return** $UVA_i$
**end**

Figure 2.7 shows an example of obtaining the UVAs. The candidate strings between the HTML markup elements are extracted to form the $FTCA_i$. Therefore, four candidate strings in $FTCA_i$ are generated. After checking the special symbols in each candidate string, seven candidate attributes in the set $UVA_i$ are available.

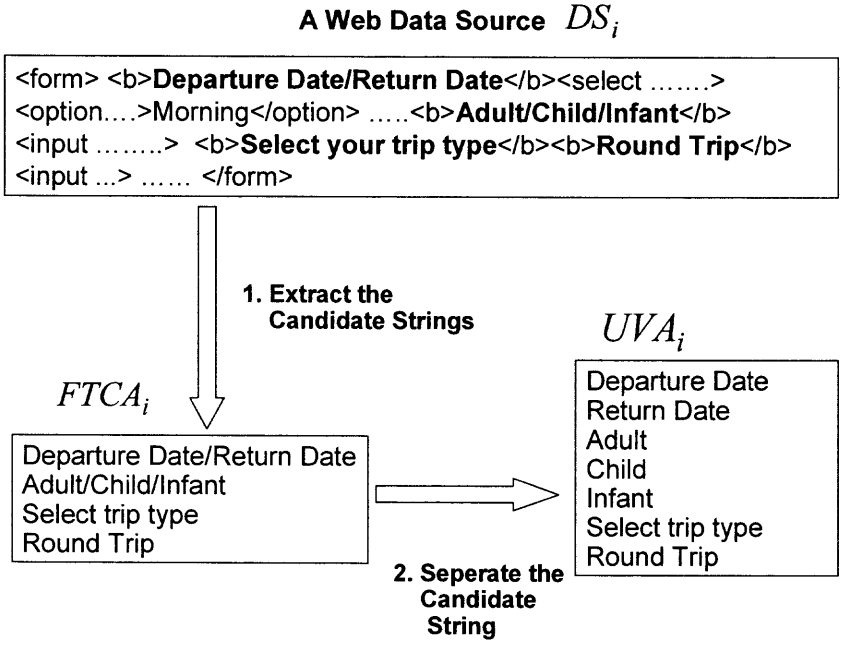



**Figure 2.7** An example of obtaining the UVA.

### 2.3.4 Final Attribute Determination Using Synonyms

This section presents the procedures to obtain the synonyms of PVA and UVA, using the WordNet ontology and to determine the final attributes.

**2.3.4.1 Ontology-based Attribute Expansion.** An ontology is employed for processing text from the query interfaces of Deep Web sources in this paper, while the Deep Web is

studied as a source for building domain ontologies in (Wu *et al.* 2005; Davulcu *et al.* 2003). In addition, an ontology is used to efficiently filter words from the Deep Web data sources. The ontology adds a semantic layer to the Deep Web.

WordNet is a lexical ontology and contains more than 166,000 words. Each word consists of a string and a corresponding sense (Miller 1995). In this paper, WordNet is utilized for finding matches between PVAs and UVAs, based on synonyms. It is also used for eliminating stop words to allow correct attribute retrieval. Among the WordNet categories, (nouns, verbs, adjectives, and adverbs), nouns and adverbs are prevalent based on the observation that semantics are mostly carried by nouns and adverbs (Varelas *et al.* 2005).

Obtaining the final attributes by comparing PVAs and UVAs involves additional difficulties. (1) Some candidate attributes in PVAs are abbreviations. For example, "dep" and "yr" are widely used to indicate "departure" and "year." (2) Some candidate attributes in UVAs appear in different forms (singular and plural) for example, "adults" and "adult."

When WordNet is used to remove improper words, two rules are used to filter the candidate attributes. First, each individual word of a candidate attribute is examined to see whether or not the word has a noun meaning in WordNet. If it has a noun meaning, the word will be kept, otherwise, the word is discarded. However, some useful words may be discarded due to the first rule. For example, "from" and "to" would be discarded since they do not have noun meanings in WordNet. In order to solve this problem, the second rule is used to keep those important words. That is, if a word does not have a noun meaning, but it is a preposition, the word will be kept.

The algorithm for obtaining the synonym for each candidate attribute of *PVA* or $UVA_i$ is shown below where *SCA* indicates the synonym of candidate attribute.

**Algorithm 2.5: Obtaining Synonyms of PVA or UVA**
**function** *PSA* (*PVA* or $UVA_i$): *SOPVA* or $SOUVA_i$;
**begin**
  **for** each candidate attribute in *PVA* or $UVA_i$ **do begin**
    Set *SCA* to be an empty string
   **if** the candidate attribute (CA) contains more than one sub words **then begin**
      **for** each sub-word in the *CA* **do**
        **if** (the sub-word has a noun meaning in WordNet) **or**
          (the sub-word has an adverb phrase in WordNet)
         **then** add the simple format of sub-word into *SCA*;

        **if** SCA is not an empty string **then** add one row into
          *SOPVA* or $SOUVA_i$ with the format of *SCA # SCA*
   **end**
    **else begin**
      **if** (the candidate attribute has a noun meaning in
        WordNet) **or** (the sub-word has a adverb phrase in
        WordNet) **then**
        **if** *CA* is plural **then begin**
          replace the *CA* by its singular format
          choose all synonym of the first sense of the CA
           from WordNet to form *SCA*, add one row into
           *SOPVA* or $SOUVA_i$ with the format of *CA # SCA*
         **end**
**end end**

Figure 2.8 is an example of performing the synonym analysis by utilizing WordNet. Each row in SOPVA or $SOUVA_i$ was separated into two sub-strings by the symbol "#." The string before "#" is the singular format of the PVA/$UVA_i$, and will be called target PVA/UVA in this proposal. The remaining strings after "#" are the synonyms of the target PVA/UVA.

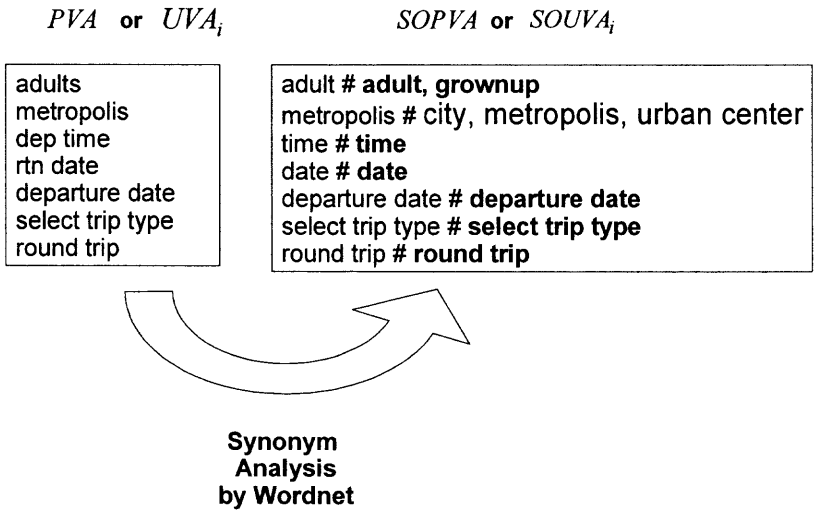



**Figure 2.8** An example of performing the synonym analysis.

**2.3.4.2 Final Attribute Extraction.** After obtaining the synonym sets, *SOPVA* and $SOUVA_i$, the final attributes are derived by comparing each row in $SOUVA_i$ to all the rows in *SOPVA*.

A two-step comparison is used to determine the final attributes by checking for overlap. First, the comparison between PVA in *SOPVA* and $UVA_i$ in $SOUVA_i$ is performed and secondly the synonyms of both sets are compared.

In the first comparison, a target UVA in $SOUVA_i$ is considered as a final attribute if there exists one target PVA in *SOPVA* such that a sub-string of the target UVA is exactly matched by the target PVA and this sub-string contains more than $\alpha$ % of all consecutive characters of the target UVA. For example, we want to determine whether a target UVA "departure date" can be considered as a final attribute or not where $\alpha$ =50%.

Suppose there exists a target PVA "departure" in *SOPVA*. Then, since the sub-string "departure" in the target UVA is larger than 50% (9/14) of the target PVA's length, the target UVA "departure date" will be considered as one of the final attributes of the Web data source.

In the second comparison, a target UVA in $SOUVA_i$ will be considered as a final attribute if one of its synonyms is exactly the same as one of the synonyms of a target PVA. For example, a target UVA "metropolis" in $SOUVA_i$ will be considered as a final attribute since one of its synonyms is exactly same as one of the synonyms of a target PCA "city" in *SOPVA*. The algorithm for final attribute extraction is presented below, and Figure 2.9 shows an example of extracting the final attributes.

```
Algorithm 2.6: Final Attribute Extraction
function FAD ( SOUVA_i , SOPVA, ): FA_i ;
begin
  for each row in SOUVA_i do begin
     Obtain the target UVA
     bContinue = True
    for each row in SOPVA do begin
      Obtain the corresponding target PVA
      /*the first comparison*/
      if the target PVA is a sub-string of the target UVA then
          if (the length of the PVA >= α % of the length of target UVA) then begin
            Add the target UVA to FA_i set; break;
          end
      /* the second comparison*/
       if one of the synonyms of the target PVA is same to one of the synonyms
         of the target UVA then begin
         Add the target UVA to FA_i set;  break;
       end
     end
  end
```

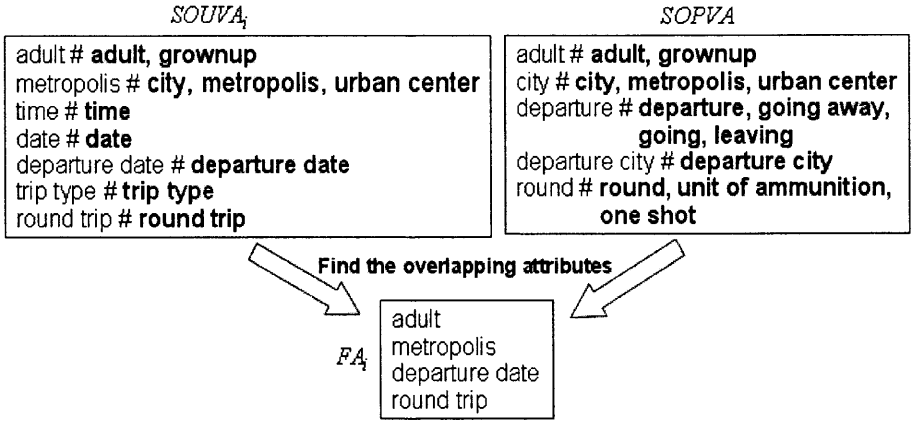



**Figure 2.9** An example of obtaining the Final Attributes.

## 2.4 Results

### Evaluation Architecture

In addition, in order to test whether the FAs can be used to identify Deep Web sources, Kabra *et al.* (2005)'s *Online Iterative Algorithm* (OIA) was implemented in this chapter. The algorithm gets user query terms and attributes and outputs a list of Web data sources in the order of the relevance to the user query. The reason of adapting OIA is that if the FAs (by the AAE algorithm) can be substituted for attributes which were selected by human experts with a perfect precision result, it can be claimed that the FAs are effective. As seen in Figure 2.10, the FAs, which are called AAE attributes, will be compared with the manual attributes retrieved for this comparison, based on the results OIA produced.

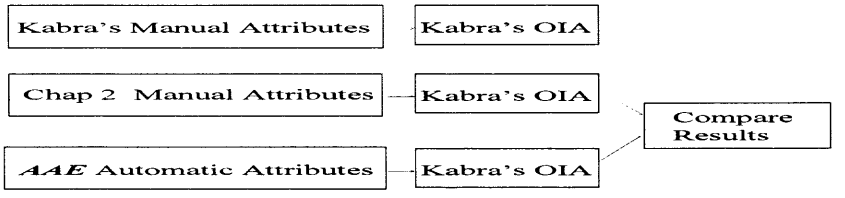



**Figure 2.10** Evaluation architecture.

Kabra *et al.* (2005) presented an *attribute co-occurrence* matrix representing frequencies of co-occurring attributes over the different Web data sources. This matrix is passed to an algorithm which ranks Web data sources. Their algorithm, consisting of three main steps (Equation 2.1, 2.2 and 2.3) is employed in this dissertation.

The *attribute co-occurrence* matrix, $w_{(a_i,a_j)}$, is computed, based on co-occurrences of a pair of attributes over Web data sources. Let $countSources(a_i \wedge a_j)$ denote the number of Web data sources which contain both attributes $a_i$ and $a_j$ then, the $w_{(a_i,a_j)}$ is computed as:

$$w_{(a_i,a_j)} = \frac{countSources(a_i \wedge a_j)}{\sum_{x=1}^{k}\sum_{y=i+1}^{k} countSources(a_x \wedge a_y)} \tag{2.1}$$

where $k$ is the size of a set of attributes and $a_i, a_j, a_x$, and $a_y \in \{FA_i\}$.The Final Attributes $FA_i$ are the ones derived in this dissertation.

Next, the relevance of an attribute $a_j$ to a user query denoted by $P[a_j]$ was calculated (Kabra *et al.* 2005) as:

$$P[a_j] = \sum_{a_i \in \{FA_i\} \setminus a_j} (d \times w_{(a_i,a_j)} \times P[a_i]) \tag{2.2}$$

where $d$ is the decay factor, and $0 \leq d < 1$.

Lastly, each Web data source will have a *relevance score* for the user query, denoted by $score(DS_i)$. The $score(DS_i)$ is computed by the following equation, and the higher $score(DS_i)$ is, the higher a Web data source is ranked. Note that each Deep Web data source was manually annotated by a set of key words which can be matched to attributes.

$$Score(DS_i) = \frac{\sum_{a \in FA_i} P[a]}{|FA_i|} \tag{2.3}$$

where $|FA_i|$ indicates the number of final attributes.

With aforementioned equations the OIA (Kabra *et al.* 2005) were implemented. The OIA algorithm outputs the top 30 Web data sources for each sampled user query based on $FA_i$. Its results, focused on the effectiveness of $FA_i$ will be discussed below.

**Implementation of Algorithms**

The algorithms was implemented with Borland Delphi and C++ Builder. Borland Delphi is used to automatically download the Web data sources, extract the PVAs, UVAs and FAs respectively.

The interface of the main algorithm developed in Borland Delphi is shown in Figure 2.11 and Figure 2.12. Figure 2.11 shows the interface of the module that extracts the final attributes from a set of Web data sources. The detailed explanation for the function of each button is shown below according to the AAE algorithm:

1. The Button labeled "Step 0: Load From DeepWeb" is used to load a list of Deep Web data sources. Each text contains a Deep Web data source, $DS_i$ written in HTML.

2. The Button labeled "Step 1: Form Label Analysis" is used to extract label elements from the set of HTML form elements $HF_i$

3. The Button labeled "Step 2: Obtain PVA (Auto)" is used to perform the Algorithm 2.2 (i.e.,Separating the Set of Inner Identifiers) and the Algorithm 2.3 (i.e., Obtaining PVAs).

4. The Button labeled "Step 3: Get Final Attributes (Auto, Ori)" is used to perform the Algorithm 2.4 (i.e., Obtaining UVA) and to find a string matches between PVA and UVA.

5. The Button labeled "Step 4: Synonym (for PVA, UVA)" is used to perform the Algorithm 2.5(i.e., Obtaining Synonyms of PVA or UVA).

6. The Button labeled "Step 5: Get Final Attribute (WordNet)" is used to perform the Algorithm 2.6 (i.e., Final Attribute Extraction) using synonyms.

7. The Button "Step 6: Calculate W*ij*" is used to calculate the attribute co-occurrence matrix.

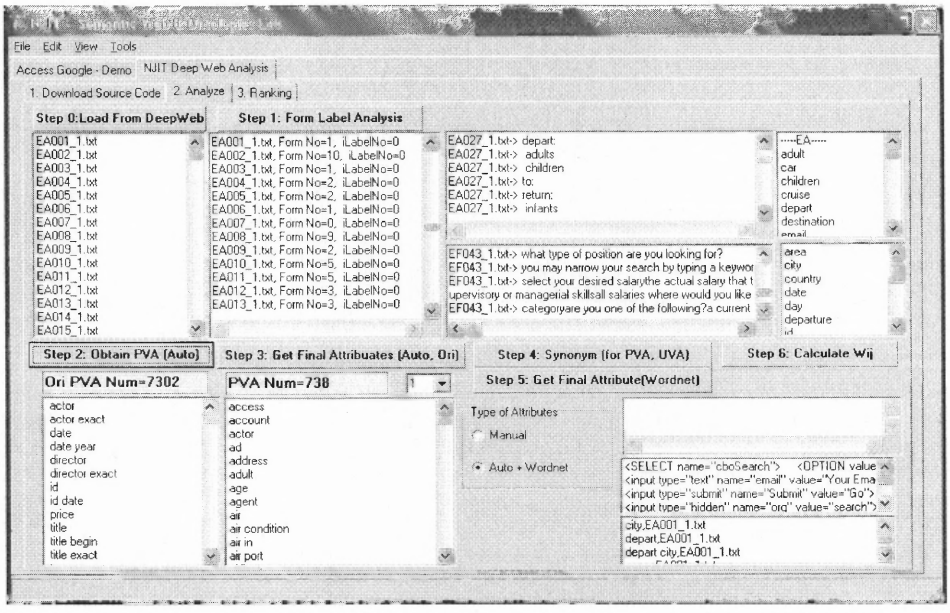



**Figure 2.11** Interface for performing AAE.

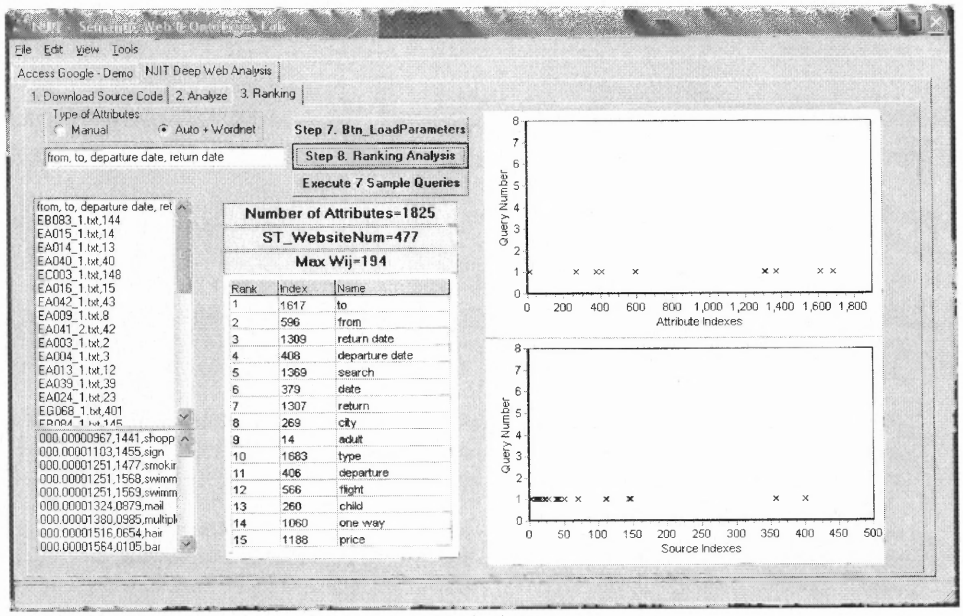



**Figure 2.12** Interface for ranking Deep Web data sources.

Figure 2.12 shows the interface of the module which compares the performances of the automatically and manually extracted final attributes. The detailed explanation for the function of each button is shown below:

1. Button "Step 7: Btn_LoadParameters" is used to load the attribute co-occurrence matrix. There is an option to choose either manually or automatically extracted attribute sets.

2. Button "Step 8: Ranking Analysis" is used to select the first 30 data sources closely relative to a simulated user query and the first 15 attributes in the order of $P[a_j]$.

In order to access the WordNet database, an interface developed in C++ Builder was developed, shown in Figure 2.13. Algorithm 2.5 calls the executable file to obtain the synonym information from WordNet.

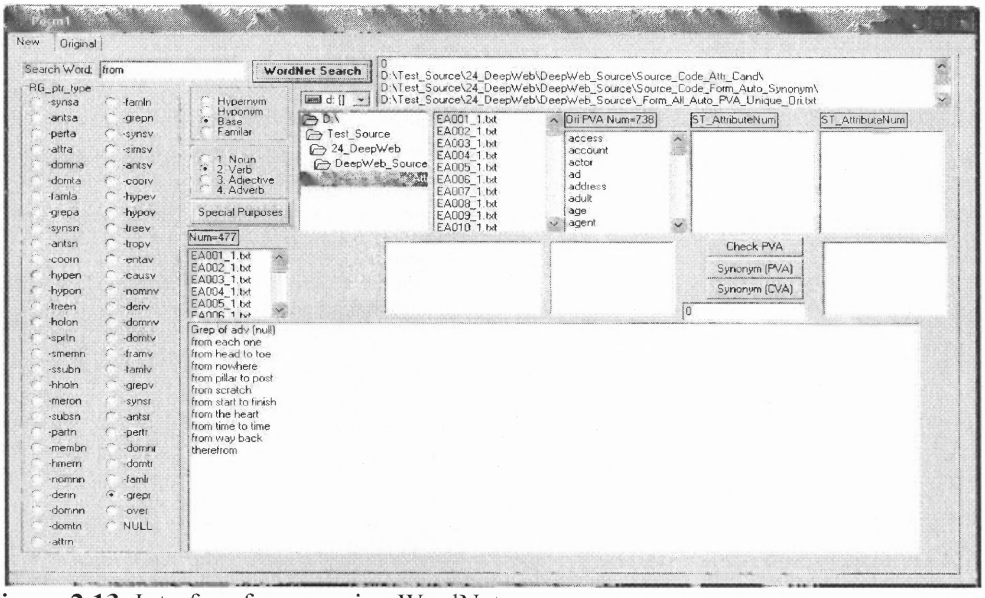



**Figure 2.13** Interface for accessing WordNet.

### Effectiveness of Attributes

In order to examine the effectiveness of attributes the automatic attribute extraction (AAE) algorithm extracted, two experimental attribute sets were generated. One was manually obtained by an experienced Web programmer, another one was automatically obtained by the AAE algorithm. They are named "manual attribute set" and "automatic attribute set" afterward in this dissertation. The Deep Web data sources were downloaded from the UIUC Web integration repository (Chang *et al.*, 2003) which contains 477 Web data sources in 8 domains, airfares (49 documents), automobiles (97), books (67), car rentals (25), hotels (39), jobs (52), movies (78), and music records (70). Figure 2.14 shows an example of manual and automatic attribute sets extracted from one of the Deep Web data sources in (Chang *et al.*, 2003). There are three query interfaces in the Web data source, and the sizes of automatic and manual attribute sets are 28 and 17, respectively.

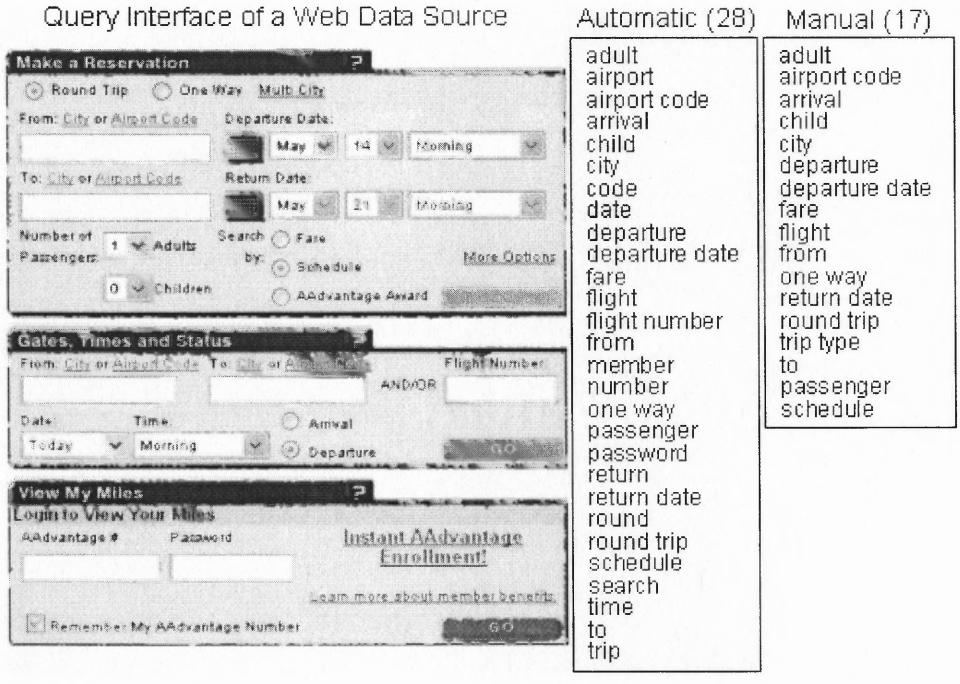



**Figure 2.14** An example of automatic and manual attribute sets obtained from the query interface of a Web data source.

The seven test queries used in this dissertation are from Kabra *et al.*'s paper (2005) as follows:

| | |
|---|---|
| Query 1 | (from, to, departure date, return date) |
| Query 2 | (author, title, ISBN) |
| Query 3 | (make, model, price) |
| Query 4 | (song, album, artist) |
| Query 5 | (title, actor, director) |
| Query 6 | (from, to) |
| Query 7 | (from, to, adult, child) |

The relationships between each query and its corresponding query domain are (Query 1, airfare), (Query 2, books), (Query 3, automobiles), (Query 4, music records), (Query 5, movies), (Query 6, airfares), and (Query 7, airfares).

Measuring how well the AAE algorithm works, compared with a human, depended on a set of keywords of a user query. The rate of correctly found Deep Web sites relevant to a user's interests, called *precision* (Hawk 1999), is regarded as a measure for the effectiveness of the attributes that the AAE extracted.

Table 2.1 shows precision and recall values for the top-30 query results in both attribute sets in Queries 1 – 7. Precision and Recall were computed as follows :

$$\text{Precision} = \frac{\left|\{relevantWebDataSources\} \cap \{retrievedWebDataSources\}\right|}{\left|\{retrievedWebDataSources\}\right|} \tag{2.4}$$

Note that *retrievedWebDataSources* is the number of Web data sources returned by Kabra *et al.*'s ranking algorithm and fixed to the 30-top sources. The *relevantWebDataSources* is the number of Web data sources which are properly matched to a sampled query.

$$\text{Recall} = \frac{\left|\{relevantWebDataSources\} \cap \{retrievedWebDataSources\}\right|}{\left|\{TotalrelevantWebDataSources\}\right|} \tag{2.5}$$

Note that the *TotalRelevantWebDataSources* is the total number of Deep Web sites which belong to a certain domain e.g., *TotalRelevantWebDataSources* for the airfare domain is 49 in the UIUC Web integration repository (Chang *et al.* 2003) which is the test data set used in this chapter. The experts (Chang *et al.* 2003) manually assigned every website in the test data set to a domain.

The differences between the manual set of attributes and the automatic set of attributes were small, with a range of 0 to 0.27 precision difference for the top-30 sources retrieved. The "manual attribute set" shows 0.96 average precision while the "automatic attribute set" shows 0.74 average precision.

**Table 2.1** Quality of Top-30 Querying Results in Both Attribute Sets in *Query 1 – 7*.

| | **Manual Attribute Set** | | | **Automatic Attribute Set** | | |
|---|---|---|---|---|---|---|
| | No. of Relevant Web data sources | Precision | Recall | No. of Relevant Web data sources | Precision | Recall |
| Query 1 (Airfare) | 30 | 1.00 | 0.61 | 24 | 0.80 | 0.49 |
| Query 2 (Books) | 30 | 1.00 | 0.45 | 21 | 0.70 | 0.31 |
| Query 3 (Automobiles) | 30 | 1.00 | 0.30 | 30 | 1.00 | 0.31 |
| Query 4 (Music) | 27 | 0.90 | 0.39 | 23 | 0.77 | 0.33 |
| Query5 (Movies) | 28 | 0.93 | 0.36 | 20 | 0.67 | 0.26 |
| Query 6 (Airfare) | 28 | 0.93 | 0.57 | 18 | 0.60 | 0.37 |
| Query 7 (Airfare) | 30 | 1 | 0.61 | 19 | 0.63 | 0.39 |

As another comparison technique, the ranking method from Kabra *et al.*'s paper (2005), as shown in Figure 2.15, which takes a set of attributes as an input, can be utilized for comparing two different sets of attributes. As mentioned earlier, the set of attributes automatically generated by the algorithms presented in this chapter can be compared with the set of attributes manually selected by experts. This method has the advantage that it is based on visualization.

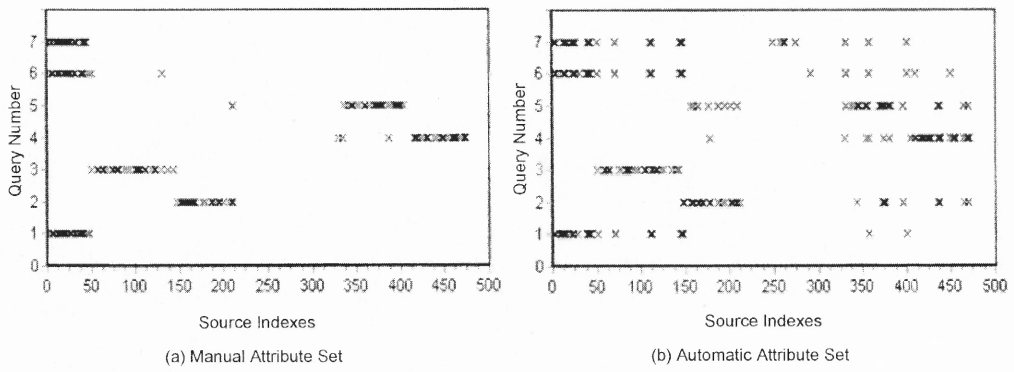



**Figure 2.15** The top-30 sources in two different attribute sets.

The *X* axis represents the Web data source indices, i.e., indexed Web sites. The *Y* axis represents the queries. It plots the top-30 sources for each test query for both manual and automatic attribute sets, respectively. Note that, the Web data sources are indexed according to their domain groups: airfares (sources 0 – 48), automobiles (49 – 145), books (146 – 212), car rentals (213 – 237), hotels (238 – 276), jobs (277 – 328), movies (329 – 406), and music records (407 – 476).

Hence, if the plot points × for a certain query number appear together within the Web source indices of the same domain group, the given attribute set can be claimed to be useful for locating relevant Web data sources with respect to the query. If the plot points × for a certain query number appear randomly in the Web source indices of the different domain groups, the given attribute set is less useful for locating relevant Web data sources with respect to the query.

The manual attribute set in Figure 2.15(a) is analyzed first. For Query 1, 2 and 3, all the top- 30 sources are matched to the corresponding domains. That is, the 30 × are consecutive in a certain range of the source indexes. For example, Query 3 is about automobiles and the index numbers of Web data sources which are ranked as being

highly relevant to this query should be between 49 and 145. The 30 × are visible in this range. However, for Query 5 the domain of which is movies, one × which represents a Web data source fell into the domain of books. This case makes the range of source indexes for movies overlap with the range of source indexes for books by the one ×

In the analysis of the automatic attribute set in Figure 2.15(b), the results in Queries 1 – 7 seem to show that the selected Web data sources are related to the clients' interests. However, they are not as good as the manually extracted attribute set since there are some overlapping sources in different queries. It is not surprising to get these results, due to the following reasons. First, the final attribute set of each Web data source is obtained by comparing each $UVA_i$ in $SOUVA_i$ to each PVA in SOPVA, as shown in Figures 2.5 and 2.9. Therefore, it is possible to get some improper attributes for the Web data sources. For example, "from" and "to" are the two best-match attributes to the Web data sources in the airfare domain. However, they may appear in the domains of automobiles and books because the condition "price range from $X to $Y" sometimes appeared in the query interfaces. Secondly, the score of a Web data source is highly associated with the size of the final attribute set of a Web data source. Therefore, it is possible that the size of the final attribute set of a Web data source is large, due to the automatic attribute determination strategy. That is, there are some improper attributes selected by the AAE algorithm. Therefore, the results of the automatic attribute set (with a size of 1825) are not as good as the manual attribute set (with a size of 526) in the imprecise and incomplete querying condition.

| Rank | Index | Name |
|---|---|---|
| 1 | 402 | return date |
| 2 | 138 | departure date |
| 3 | 478 | to |
| 4 | 205 | from |
| 5 | 130 | date |
| 6 | 136 | departure |
| 7 | 401 | return |
| 8 | 5 | adult |
| 9 | 494 | type |
| 10 | 490 | trip |
| 11 | 491 | trip type |
| 12 | 321 | one way |
| 13 | 197 | flight |
| 14 | 86 | child |
| 15 | 15 | airline |

(a) Manual Attribute Set

| Rank | Index | Name |
|---|---|---|
| 1 | 1617 | to |
| 2 | 596 | from |
| 3 | 1309 | return date |
| 4 | 408 | departure date |
| 5 | 1369 | search |
| 6 | 379 | date |
| 7 | 1307 | return |
| 8 | 269 | city |
| 9 | 14 | adult |
| 10 | 1683 | type |
| 11 | 406 | departure |
| 12 | 566 | flight |
| 13 | 260 | child |
| 14 | 1060 | one way |
| 15 | 1188 | price |

(b) Automatic Attribute Set

**Figure 2.16** The top-15 attributes with both attribute sets for Query 1.

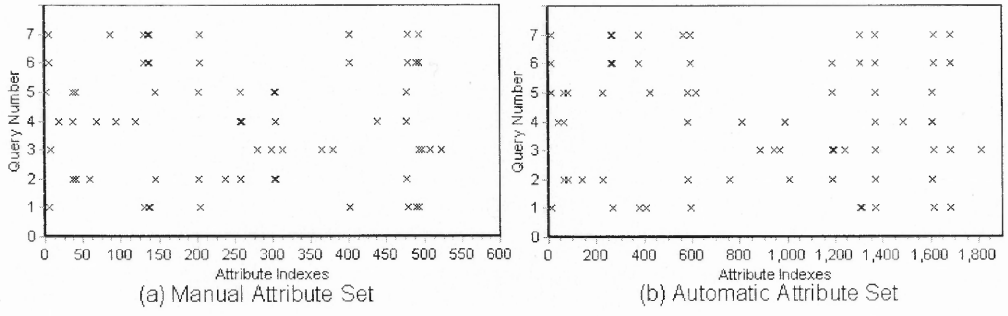



**Figure 2.17** The top-10 attributes in different attribute sets.

When a user enters a complete query like Query 1, 87% of attributes among the top-15 attributes are the same in the two attribute sets, as shown in Figure 2.16. Figure 2.17 shows the top-10 attributes that are associated with Queries 1 – 7 in both automatic and manual attribute sets.

Based on the results of comparison tests conducted, it can be concluded that the AAE can automatically extract attributes which can be used in identifying Web data sources in a similar way as human experts do. The evaluation of automatically extracted

attributes will be further considered, focusing on domain ontologies in Chapter 5 and the utility in Deep Web search, together with better assessment tests, in Chapter 4.

In conclusion, a new algorithm which can work like an expert Web programmer in extracting attributes from the query interfaces of the Deep Web by reconciling PVAs with UVAs was developed in this chapter. As a result, 1,825 final attributes from 477 Web data sources (Chang *et al.* 2003) in eight domains (airfares, automobiles, books, car rentals, hotels, jobs, movies and music records) were automatically extracted.

A manual review of the AAE algorithm showed that the results were sensible. However, one needs to keep in mind that the test data sets were relatively small, compared to the numbers of sites available in these domains on the Web. Thus, for larger sets of Web sites, a human review of the algorithm results will be necessary. The attribute extraction for realistic domains should therefore be considered as a semi-automatic process.

# CHAPTER 3

# AUTOMATIC GENERATION OF ONTOLOGY FROM THE DEEP WEB

## 3.1 Introduction

Chapter 3 introduces a novel approach to the automatic generation of domain specific ontologies by analyzing Deep Web sources. An approach and an algorithm of automatic ontology construction for Deep Web pages to enhance the representation of the Deep Web's source contents will be presented. This process is performed after extracting attributes of 447 Deep Web data sources, described in Chapter 2. The domain ontologies based on these attributes from Deep Web, are considered as features of the Semantic Deep Web in this dissertation work.[1]

Finding relevant e-commerce sites and accessing, retrieving and maintaining the huge amount of existing Deep Web information raise challenging research issues. Most of the Deep Web can only be accessed through dynamic query interfaces, which contain HTML form elements. For this, identifying a relevant rich set of keywords to represent the Deep Web contents of a site is crucial, since a wrong or sparse set of keywords may result in many irrelevant search results, thus forcing users to sift through them to find the pages that match their interests. Identifying a rich set of keywords for the Deep Web is also needed to overcome the bias of the current search engines towards popular sites, based on a link-based analysis of the sites. The link-based indexing of Web sites retrieves popular sites, but not necessarily the sites satisfying the user's service or information needs.

---

[1] This chapter's contents were published in (An *et al.* 2007a).

The Web form interface pages are considered as Deep Web services that indirectly reflect the real content types of the Deep Web. A set of attributes was automatically extracted from such interface pages in (An *et al.* 2007a). On the other hand, annotating (indexing) these Deep Web services with rich semantic concepts (keywords) will yield Web search results which include these service pages, thus facilitating access to the Deep Web contents.

To achieve this, Semantic Web technology, specifically ontologies, to annotate the Deep Web pages are utilized. Ontologies have been used to annotate Web services (Sabou *et al.* 2005; Patil *et al.* 2005; Heß and Kushmerick 2004), which support Web agents' automation of tasks such as composing Web services and customizing a workflow of services. However, constructing an ontology, i.e., a machine readable form of human knowledge, automatically from textual input remains a grand challenge. Web ontologies are needed to fulfill the vision of the Semantic Web, which is the automation of tasks on the Web that currently require human effort (Dou *et al.* 2005).

Because of ontologies' promise of sharing knowledge among humans and programs, ontologies, especially domain specific ontologies in e-commerce, will play a control role on the Semantic Web (Dou *et al.* 2005) and the Deep Web. In order to support the Semantic Web vision, the building of many domain specific ontologies is desirable. However, building ontologies is difficult, time-consuming and error-prone. Rather than using a manual approach, automatic generation of ontologies from Web sources would be a better approach, but this is a well-known difficult task (Wu *et al.* 2005).

The use of ontologies will help to recognize user interests and address them

correctly. For example, in the book sales domain, "From" means the lower end of a price range, while it usually means andeparture city (or date) in the travel domain. This shows that the content of an ontology, even if it is limited to e-commerce, needs to be domain-specific. In Chapter 3, a novel approach to the automatic generation of domain specific ontologies is presented. It is intended that the domain ontologies derived from the Deep Web should convey domain characteristics of Web pages accessing the Deep Web.

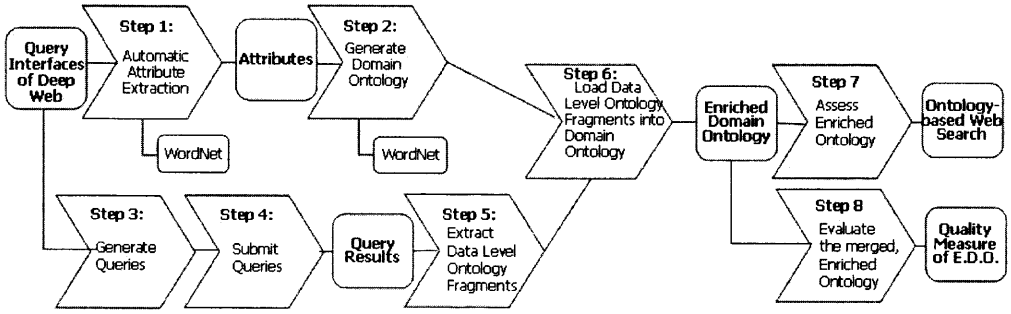



**Figure 3.1** The framework for generating domain ontologies.

Chapter 3 is about Step2 of the grand system architecture in Figure 3.1. The rest of this chapter contains sections as follows. In section 3.1, related work is presented. Section 3.2 shows the algorithm for automatically generating a domain ontology from the Deep Web. Finally, section 3.4 presents how the algorithm was implemented and its experimental results.

## 3.2 Related Work

Building an ontology automatically from Web pages has been attempted in (Davulcu *et al.* 2005; Wu *et al.* 2005; Sabou *et al.* 2005; Roitman and Gal 2006). Two approaches to build an ontology are either building from scratch or building by reusing existing

ontologies (Pinto and Martins 2004). Approaches in (Davulcu *et al.* 2005; Wu *et al.* 2005; Sabou *et al.* 2005; Roitman and Gal 2006) are examples of building ontologies from scratch, with the adaptation of some existing Artificial Intelligence methods. OntoMiner (Davulcu *et al.* 2005) can derive XML hierarchical semantic tree structures from Web sites to find concepts by using a partitioning algorithm. In order to mine IS-A relationships among the concepts (Davulcu *et al.* 2005), their algorithm uses frequency information of how strongly a pair of concepts is linked in the sub-structures of a hierarchical tree which assembles the XML semantic trees of each Web site. Schema learning (Nestorov *et al.* 1998; Garofalakis *et al.* 2000) was presented as a background study in culling a structure out of a Web site.

A DOM tree corresponding to the schema of a Web site was considered for building the ontology in (Wu *et al.* 2005; Roitman and Gal 2006). OntoBuilder (Roitman and Gal 2006) turns a Web site into a hierarchical structure on the fly and performs matching between ontologies which correspond to each Web site. DeepMiner (Wu *et al.* 2005) also uses the DOM tree in extracting concepts and instances from Web sites, based on the relative positions of the concepts in the tree. Compared with (Davulcu *et al.* 2005), the IS-A relationship is not specified in (Wu *et al.* 2005), but a clustering algorithm to explore new concepts over interfaces of Web sites is described for a domain-specific ontology. The common feature of (Davulcu *et al.* 2005; Wu *et al.* 2005; Roitman and Gal 2006) is that methodologies to bootstrap an ontology from the Web are discussed without presenting a final ontology. Thereby, scalability of the methodologies to a considerable number of Web sites is willing to be an issue.

In this dissertation, a novel approach to the automatic generation of domain

specific ontologies is presented. Contributions of the proposed approach are twofold: improved processing of users' queries (Handschuh 2003) and ontology usage for semantic annotation of Web services, as will be discussed now. First, Web sites themselves provide an environment for users to learn domain specific terms, which implicitly reflect users' understanding of the domain. Therefore, extracting frequently used concepts across many Deep Web sites implies that the domain ontologies of these sites exhibit an important feature, domain consensus, which makes the ontologies usable (Missikoff 2002). Next, the newly proposed approach to generating a domain ontology can be used for semantic annotation of Web services, since mapping service descriptions to concepts of the ontologies is essential for the discovery of services (Patil 2004).

### 3.3 Generating a Domain Ontology from the Deep Web

A domain ontology is generated, based on the automatically obtained final attributes (Chapter 2), by the following substeps. In the first substep, duplicates will be removed from the final attributes obtained by the Automatic Attribute Extraction algorithm for each of the eight subdomains (see below). Then, a superclass or parent of each attribute will be derived from WordNet. Here "attributes" refers to as the Web page attributes, as before. However these attributes appear as concepts in WordNet. In addition, "hypernym," which is the term used in WordNet, is referred to as "superclass," or "parent" in this dissertation. The final attributes are assembled into several groups of concepts which form DAGs (directed acyclic graphs) based on their nearest common ancestors. Let's call these small DAGs *schema fragments* (SFs). Note that each of the SFs contains at least one vertex, and the roots of all SFs are final attributes. The domain

ontology will be generated by iteratively merging the SFs into a final single DAG. This process terminates when only one DAG is left or a certain maximum number of iterations has been reached. Note that in each iteration, the first hypernym of a root becomes the new root of the SF. Figures 3.2 and 3.3 show examples of generating subgroups based on nearest common ancestors in the airfare subdomain.

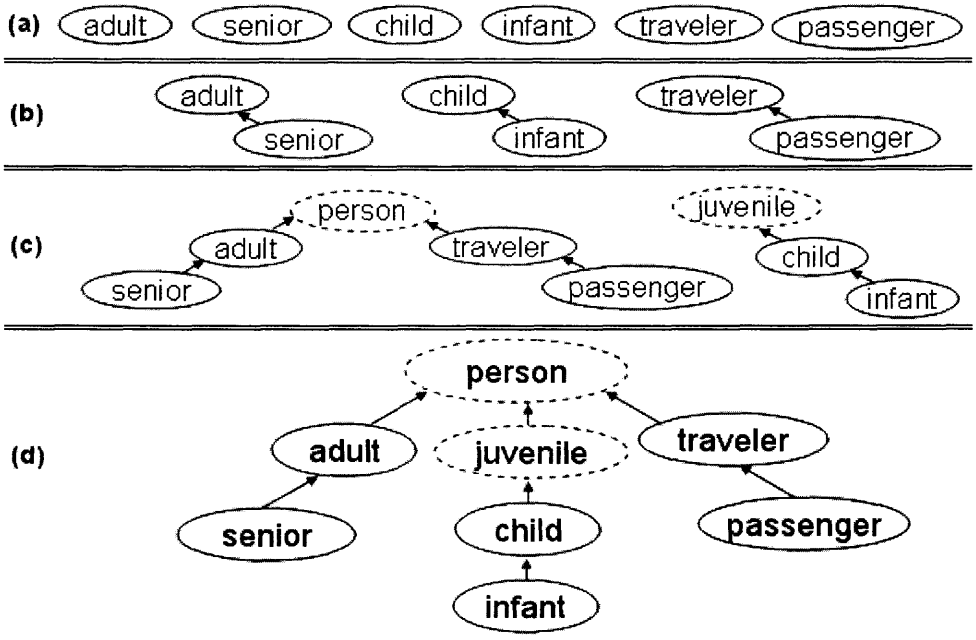



**Figure 3.2** An example of generating a SF with common nearest ancestor "person".

In Figure 3.2, there are six final attributes (adult, child, infant, passenger, senior, traveler) obtained by the AAE algorithm (An *et al.* 2007a). By retrieving their hypernyms from WordNet, it is found that those attributes can be presented by a DAG whose root is "person." Note that, the concept "juvenile" is derived from WordNet to create a

connection between "child" and "person." Figure 3.2(a) shows the six final attributes. Figure 3.2(b) shows three SFs constructed from the six final attributes. Figure 3.2(c) shows the first iteration of adding a new root. The new root "person" is the hypernym of the root in the previous iteration and comes from WordNet. The final SF of the 6 attributes is generated in the second iteration as shown in Figure 3.2(d). This is done by finding that person is a hypernym of juvenile in WordNet and adding an IS-A connection between them. A dashed ellipse stands for a concept derived from WordNet. Because WordNet was already used when deriving final attributes, it is guaranteed that the root of each SF will be found in WordNet.

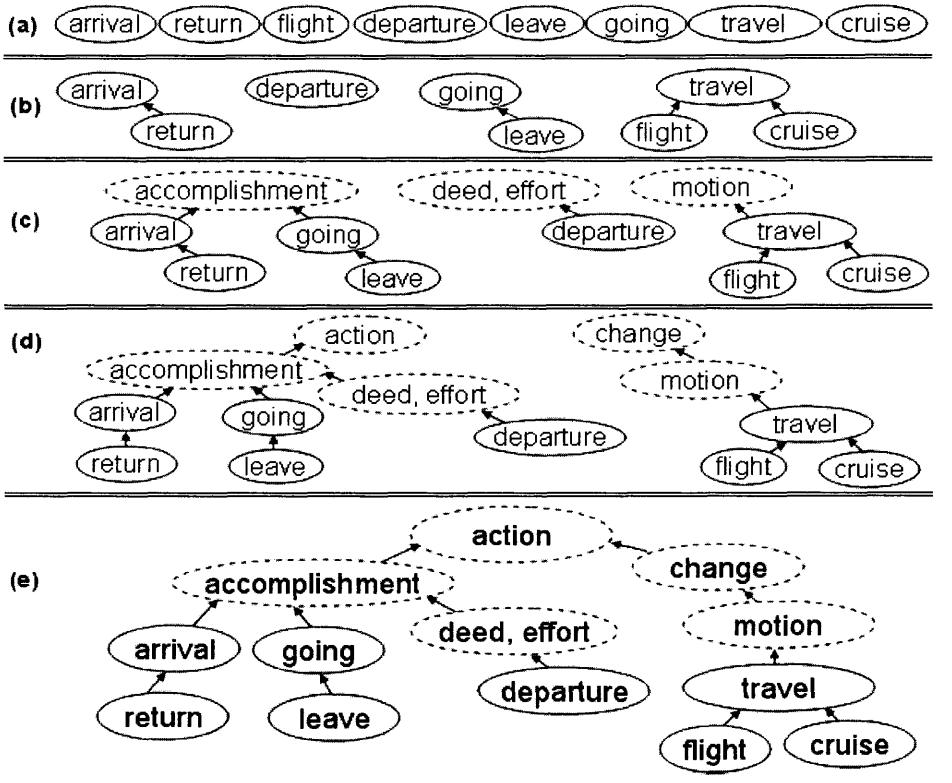



**Figure 3.3** An example of generating a Schema Fragment with common ancestor "action".

Figure 3.3 shows another example of generating a domain ontology. The eight attributes (arrival, departure, return, leave, going, travel, cruise, flight) obtained by the automatic extraction algorithm (Figure 3.3(a)) are taxonomically interwoven by examining hypernyms, and "action" is found as the nearest common ancestor of all these attributes in WordNet. Figure 3.3(b) shows four schema fragments in which the root of each schema fragment is again one of the eight final attributes. Note that, the direct hypernym of "return" is "arrival" in WordNet. During each iteration, hypernyms are added to each schema fragment as new roots as shown in Figures 3.3(c) and (d). The final schema fragment of the eight attributes is generated in the third iteration, as shown in Figure 3.3(e).

A function for obtaining the domain ontology (called *ODO*) is shown in Algorithm 3.1. Following below are the descriptions of functions used in the algorithm.

**Algorithm 3.1: Obtaining Domain Ontology**

```
function ODO(FA: set of final attributes represented as graph nodes) returns DAG
{
  //Phase 1 (Initial Schema Fragment Generation):
   forest = { }; // forest is a set of DAGs and |forest| is the number of DAGs in it
   tempDAG; // tempDAG is a variable which temporarily holds a newly created DAG
   initForest (FA);
  for (all pairs (i, j), i ∈ FA, j ∈ FA, i ≠ j ) do{
    if (is-child in WordNet(i, j)){
      tempDAG = insertIS-Alink(i, j);
       forest = forest ∪ {tempDAG};
      delete vertexes i and j from the forest;
    }else if (is-child in WordNet(j, i)) {
      tempDAG = insertIS-A link(j, i);
```

```
    forest = forest ∪ {tempDAG};
    delete vertexes i and j from the forest;  } }

//Phase 2 (Schema Fragment Merge):
 iterationCounter = 1;
const threshold; // indicates a certain number of iterations
 while ( |forest| ≥ 1 ) and (iterationCounter< threshold) do {
  for each DAGi do { DAGk = addIS-Alink(DAGi); }
   // DAGi is a DAG where i is the root of DAGi
   // k ∈ WordNet and k ∉ FA
  for (all pairs (DAGi, DAGj), i ≠ j ) do {
   if ((m = find-parent in WordNetDAG(DAGi, j) is not null) {
    // m is a parent of j where m ∈ DAG_i
       DAGn = mergeDAGs(DAGi, m, DAGj) where l = k or l ≠ k;
       forest = forest ∪ {DAG_n}
       forest = forest \ {DAG_i, DAG_j}
      } else if (m =find-parent in WordNetDAG(DAGj, i)) {
       // m is a parent of i where m ∈ DAG_j
       DAGn = mergeDAGs(DAGj, m, DAGi);
       forest = forest ∪ {DAG_n}
       forest = forest \ {DAG_i, DAG_j} } }
  iterationCounter= iterationCounter+1;
 } // end of while
}
```

- initForest(*FA*): used to initialize a forest with Unique Final Attribute(s).
  is-child in WordNet(*i*, *j*): used to find an IS-A relationship in WordNet. It return true if *i* is a hypernym of *j* in WordNet, otherwise, it returns false.

- insertIS-Alink(*i*, *j*): used to create a DAG by inserting a new IS-A link between *j* and *i*. The direction of the edge between *j* and *i* is from *j* to *i*.

- addIS-Alink(DAG*i*): used to add a new root to DAG*i* (i.e., DAG where *i* is the root) by inserting the IS-A link from *i* to its hypernym *k*, retrieved from WordNet. It returns a newly created DAG*k* where *k* is the root.

- find-parent in WordNetDAG(DAG*i*, *j*): used to find an IS-A relationship between vertex *j* and each vertex of DAG*i* in WordNet. It returns a vertex *m* if the IS-A relationship exists between *m* and *j* (i.e., *m* is a parent of *j* in WordNet where $m \in DAG_i$).

- mergeDAGs(DAG*i*, m, DAG*j*): used to merge DAG*i* and DAG*j* by inserting an IS-A relationship between the root *j* of DAG*j* and a vertex *m* where *m* belongs to DAG*i*. It returns a DAG*n* where *n* = *i* and *m* are children of *j*.

The algorithm consists of two phases (Figure 3.4): initial schema fragment generation and schema fragment merge. In the initial schema fragment generation, several initial schema fragments shaped as DAGs are generated based on the nearest common ancestors of all attributes by referring to hypernym links in Wordnet. If one of the attributes is a superclass of another attribute in WordNet, an IS-A relationship between them is inserted. The result is a new DAG, consisting of two vertexes and the IS-A link. The child of the IS-A link is removed from the set of DAGs. The parent becomes the root of the new DAG and is also removed. Note that, in this phase, the roots of all the SFs must be attributes.

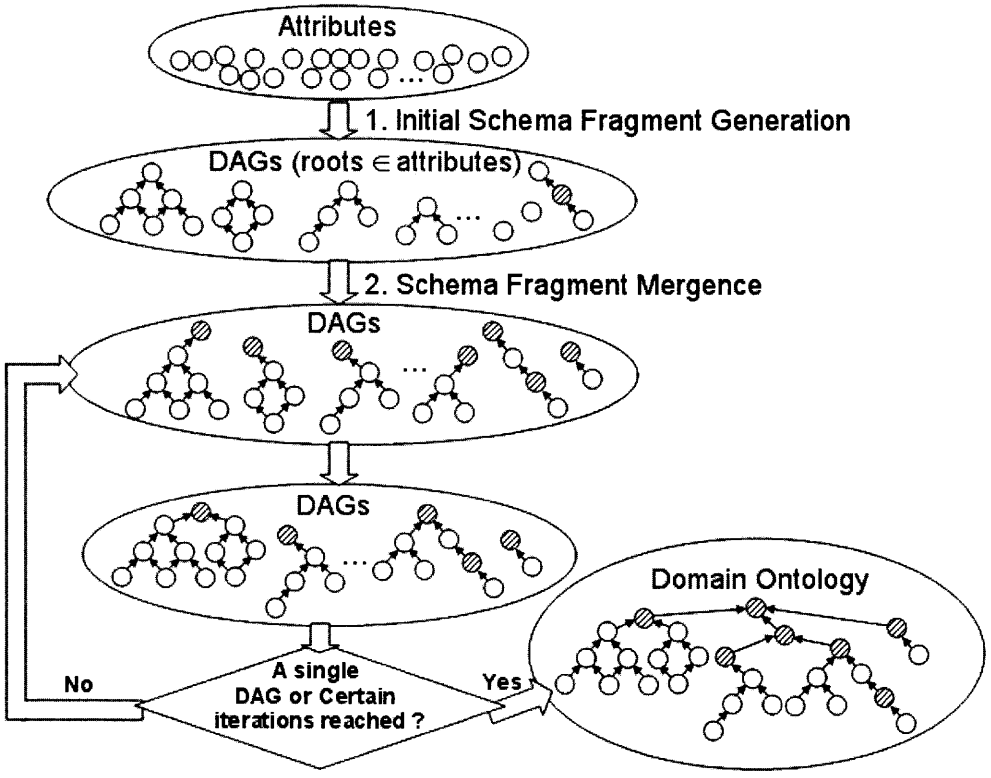



**Figure 3.4** The process of the domain ontology generation algorithm.

In the schema fragment merge phase, the algorithm relies on repeated calls to a function *merge*DAGs. This process of merging DAGs is continued by trying to locate an IS-A link (in WordNet) from the root vertex of DAG*i* to a node in DAG*j*. If this is successful, a new combined DAG is created and it contains all nodes of both and a new IS-A link connecting them. DAG*i* and DAG*j* are deleted. This process is continued as long as links are found in WordNet. Ideally, the result of this process consists of one or a few large DAGs. If no common concepts are found in a pair of DAGs, a new root is added to be the parent of the two roots (found in WordNet). For an example in Figure 3.3 (c), the DAGs rooted in "arrival" and "going" have no nodes in common. Thus,

"accomplishment" is added to be a new root above them. This process continues until either a merge becomes possible, or the WordNet root "entity" is found.

## 3.4 Results

This section demonstrates how the algorithm is implemented with its experimental results for the automatic domain ontology generation. Therefore, the automatically extracted attribute sets in each of the eight subdomains are considered as seeds for the ontology construction, with WordNet providing the missing links.

### Implementation of the Algorithm and GUI

The algorithm is implemented in Borland Delphi and C++ Builder. Borland Delphi is used to automatically download the Web data sources, extract the PVAs and UVAs and generate the domain ontologies. The graphical user interface (GUI) of the main algorithm (Algorithm 3.1: Obtaining Domain Ontology) developed in Delphi is shown in Figure 3.5.

Here is a succinct guide for the GUI to generate a domain ontology. First, a user needs to load the attributes which were automatically extracted in Chapter 2 in the interface by indicating a directory path to access them. The button labeled "Schema Level – 1st Level" performs the Phase 1 (i.e., Initial Schema Fragment Generation) of Algorithm 3.1 and the button labeled "Schema Level – Final" performs the Phase 2 (i.e., Schema Fragment Merge) of Algorithm 3.1. In addition to generating a domain ontology in OWL format, the interface provides a user with a space to trace IS-A relationships for a given attribute. For example, the derived domain ontology for the "airfares" domain

contains 525 attributes that are visible information on the interface. Furthermore, an attribute "adult" among those attributes is shown to be a root of a schema fragment in Phase 1, but not a root in Phase 2 which completes generating the domain ontology. The attribute has 2 parents with index numbers "379" and "385," and has 4 children with index numbers "74," "92," "133," and "161."

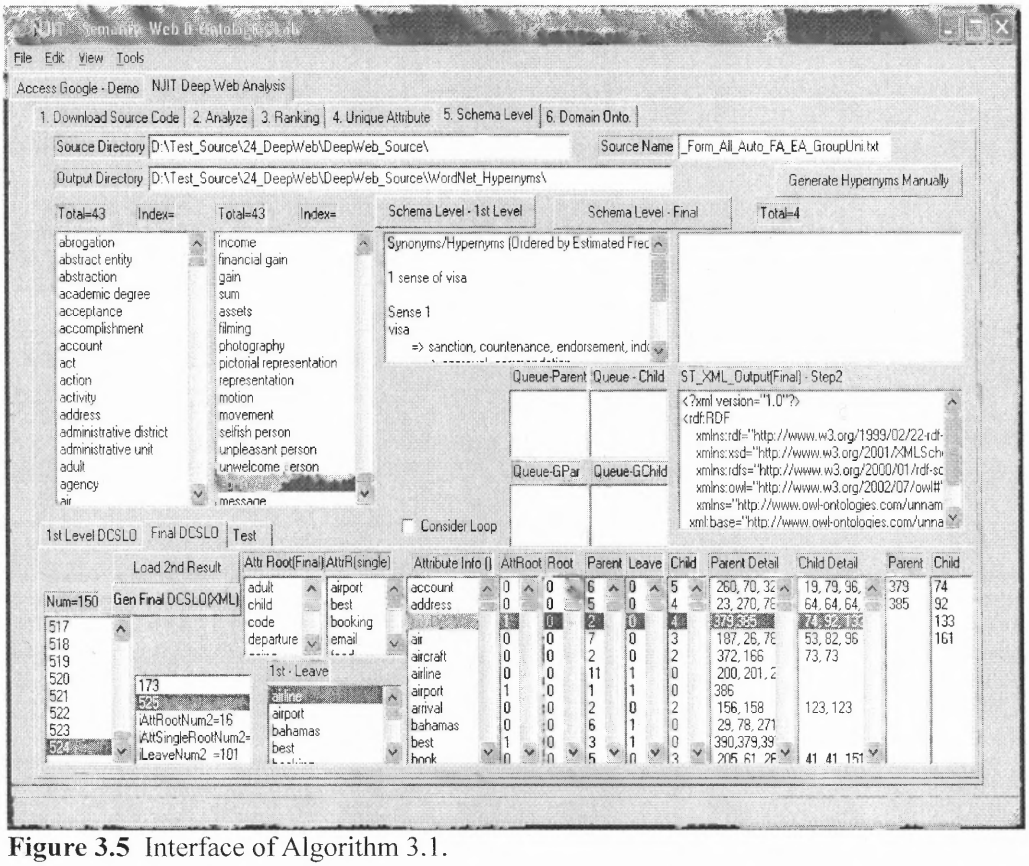



**Figure 3.5** Interface of Algorithm 3.1.

In order to access the WordNet database, the interface, developed in C++ Builder, is shown in Figure 3.6. That is, the main algorithm calls the executable file to obtain the synonym/hypernym information from WordNet.

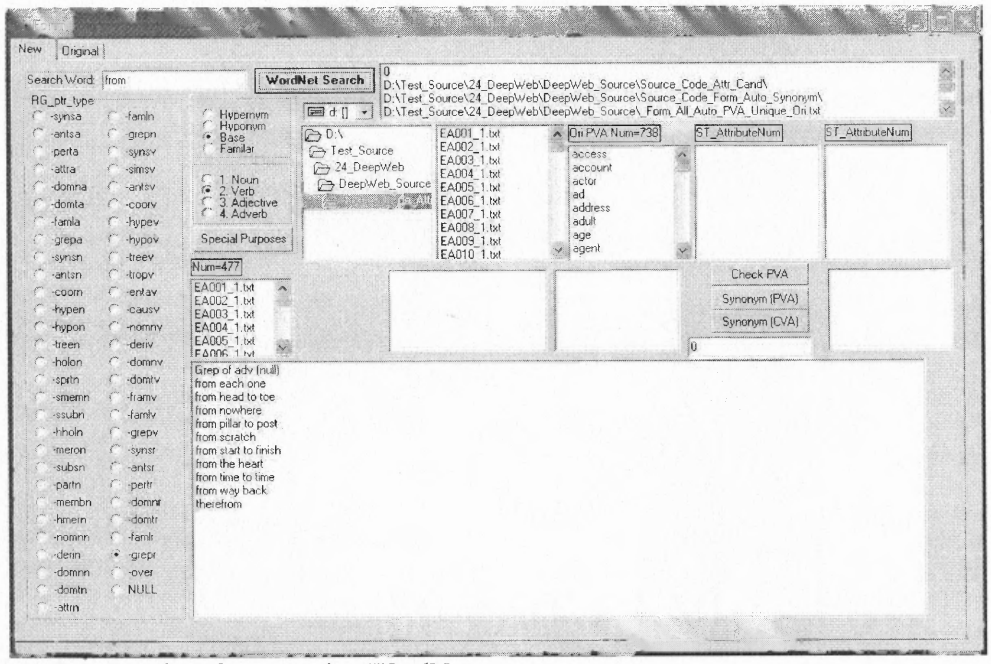



**Figure 3.6** Interface for accessing WordNet.

### Generation of Domain Ontology

As shown in Section 3.3, the ODO algorithm consists of two phases. In Figure 3.7, parts of two automatically generated schema fragments with the same common ancestors are given. The sample concepts shown in Figures 3.3 and 3.4 are marked by the ellipses in Figure 3.7.

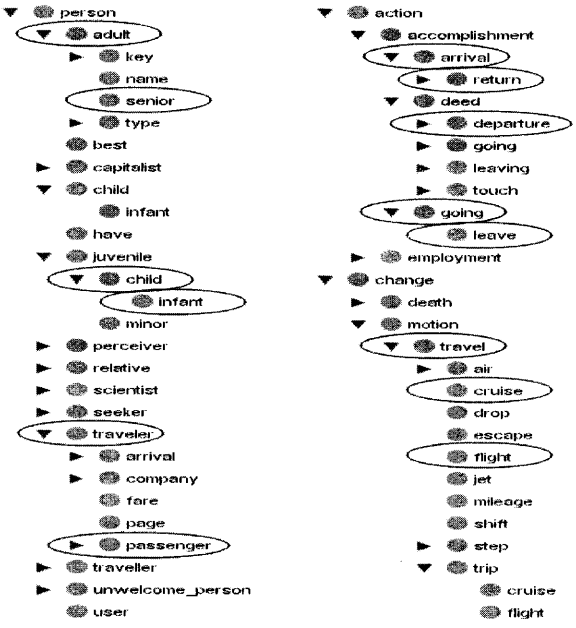



**Figure 3.7** The SFs of common ancestors "person" and "action," respectively.

**Table 3.1** Size Information after First Phase of the ODO Algorithm

| | Number of concepts | SFs | SFs (One node) | Leaves | Added from WordNet |
|---|---|---|---|---|---|
| **Airfares** | 173 | 16 | 27 | 74 | 379 |
| **Automobiles** | 212 | 11 | 23 | 105 | 582 |
| **Books** | 220 | 11 | 30 | 101 | 558 |
| **Car Rentals** | 120 | 14 | 14 | 50 | 323 |
| **Hotels** | 311 | 15 | 32 | 158 | 811 |
| **Jobs** | 306 | 3 | 31 | 151 | 880 |
| **Movies** | 193 | 2 | 15 | 91 | 595 |
| **MusicRecords** | 157 | 16 | 27 | 69 | 306 |

Table 3.1 shows the results for the eight subdomains after the initial schema fragment generation phase of the algorithm. The column "Number of Concepts" indicates the unique single-word attributes obtained in each automatic attribute set. If an attribute contains two words, it is separated into two new attributes. "SFs" and "SFs (One node)" indicate the total numbers of schema fragments of multiple nodes and one node, respectively. "Leaves" indicates the total number of attributes which belong to leaves of SFs. "Additional concepts from WordNet" indicates the total number of concepts obtained from WordNet in order to connect the schema fragments.

In the schema fragment merging phase of the ODO algorithm, which corresponds to the second step (i.e., iteration) in Figure 3, 146 additional concepts were taken from WordNet to form the final result, that is, the domain ontology for the Airfares domain. In addition, 120 concepts for Automobiles, 135 concepts for Books, 121 concepts for Car Rentals, 140 concepts for Hotels, 99 concepts for Jobs, 62 concepts for Movies and 216 concepts for the Music records domain, respectively, were derived from WordNet to merge all SFs to generate a domain ontology for each domain.

Concepts (i.e., attributes of the Deep Web data sources) are harvested from many Web sites and the concepts are iteratively interwoven into the IS-A skeleton (i.e., hyponym structure) of the well-known WordNet ontology. As a result, domain ontologies in the eight subdomains were generated with a considerable size and information about concepts and IS-A relationships of the concepts. The concepts in the generated ontologies do not have multiple parents in their hierarchical structures. Because the most frequently used sense of each concept was selected from WordNet in locating IS-A relationships between concepts, the hierarchical structures of the generated domain ontologies are

trees. In the theory of algorithms, a tree is considered a special case of a directed acyclic graph (DAG). However, the theory was expressed in the most general terms used in ontologies.

In conclusion, eight complete domain ontologies were successfully automatically generated. In all, 1,825 concepts were interwoven into the IS-A relationships of WordNet and 5473 concepts directly from WordNet were included in the generating process.

One weakness of our algorithm is that it does not distinguish between (two word) phrases and other sequences of words. The reason that this is acceptable in our environment is that our post processing with WordNet would not allow the use of a two word phrase as a concept. However, this kind of limitation should be relaxed in the future. Then it will become important to recognize phrases (with two or more words) as carrying a distinctive meaning.

One aspect of significance of the presented approach is that a domain ontology is automatically built by analyzing the Web data sources. As a result, the generated domain ontology is (so far) a subset of WordNet, but it is pruned to represent the domain. More importantly, this ontology forms the foundation for adding new domain-specific concepts that do not exist in WordNet.

Another aspect of significance of this approach is that domain ontologies from many Deep Web sites will improve the processing of user queries. Unlike other research (Davulcu *et al.* 2003; Wu *et al.* 2005), this dissertation work generates ontologies in the OWL format, which can be edited with existing tools (e.g., Protégé) to be applied in Web search and other applications.

# CHAPTER 4

# INSTANCE EXTRACTION FOR ONTOLOGY-BASED DEEP WEB SEARCH

## 4.1 Introduction

In the previous Chapters 2 and 3, the search interface attributes of many Deep Web sites have been extracted to automatically build domain ontologies for the Semantic Web. The concepts extracted from Deep Web sites were iteratively interwoven into the concept hierarchy (hyponym structure) of WordNet. The focus in Chapter 3 was to build a domain ontology with carefully selected concepts in a certain domain. Chapter 4 introduces a method to extract instances from the Deep Web, especially from structured backend databases to enrich the domain ontology generated in Chapter 3.[1]

The Semantic Web is the next generation of the World-Wide Web, in which many tasks that humans perform with the WWW are automated. By automatically combining information from software agents, which find data and services, the Semantic Web satisfies more sophisticated needs of Web users. The Semantic Web depends heavily on ontologies, each of which is a computer implementation of human-like knowledge (Gruber 1995). The major components of an ontology include concepts and their instances (Davulcu 2003) and relationships between concepts. The software agents on the Semantic Web will need some human-like knowledge to perform their tasks. Thus, the success of the Semantic Web critically depends upon the existence of a sufficient amount of high-quality semantics contained in ontologies (McDowell 2006) which can be shared between application programs and humans.

---

[1] This chapter's contents were submitted for publication.

The Deep Web is a great source for extracting ontologies because of two reasons. The Deep Web contains a great amount of information, and instances from the Deep Web provide rich, high-quality semantics, based on the designed, structured data representations of the backend databases. In other words, it is easier to pin down the semantics of a data item that was derived from a relational table than for a data item coming from a free-text Web page. The ontologies enriched with Deep Web instances are then applied for Web search to improve the search results by better Deep Web sites. The search terms entered by a user, when augmented with domain ontology instances, are expected to better describe the user's interests.

Relating back to the architecture in Figure 4.1, in this chapter, in order to enhance a domain ontology, instances and new concepts based on results obtained by querying suitable Deep Web sites are added to the domain ontology. As a new Website is visited, new instances and concepts can be recognized and merged into the ontology. This chapter will also address the question how to access Deep Web data from a site which is not designed to cooperate with crawling algorithms. The newly proposed approach to this problem is as follows. Hidden instances from the Deep Web are obtained by probing an entry field and evaluating error messages returned by its front-end Web page.

It will be shown that the proposed method assists users with finding more relevant Web sites. In order to justify this claim, a domain ontology-based Web search system was implemented. The assessment was conducted by comparing the number of relevant Deep Web sites returned by a general purpose Web search engine, with a user's search terms only, as opposed to the results returned by entering search terms extended with terms from an ontology containing instances extracted from the Deep Web.

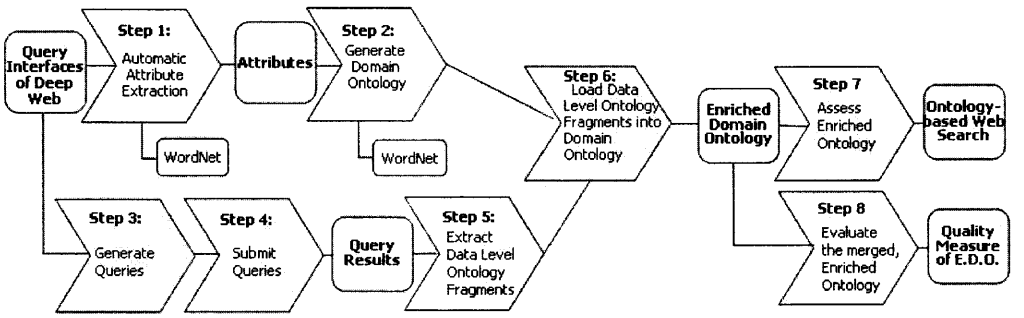



**Figure 4.1** The frame work for generating an enriched domain ontology.

Thus, chapter is about Steps 3, 4, 5, 6 and 7 of the grand system architecture (Figure 4.1). The rest of this chapter is organized as follows. Section 4.2 presents related work. Section 4.3 describes the novel approach, which enhances a domain ontology for the Semantic Deep Web (steps 3 to 6 in Figure 4.1). Section 4.4 presents an algorithm that extends the list of search terms, given by a user, with the help of the enhanced domain ontology, to obtain better search results. Finally, Section 4.5 discusses some limitations of the presented algorithm.

## 4.2 Related Work

Information in structured databases of Deep Web sites can be searched through query interfaces which ask a user to fill in proper input fields and return a corresponding result to the user. This dissertation focuses on such structured databases. Especially, this chapter addresses the problem of how to extract these structured data from the Deep Web to automatically enrich a domain ontology that can support better search engines.

Bootstrapping techniques in building ontologies have been exploited in (Davulcu

*et al.* 2005; Wu *et al.* 2005; Roitman and Gal 2006 ). On the other hand, visual layouts of form elements and the sequence of forms across pages in a Web site may convey some implied semantics but the depth and width of semantics inferred from query interfaces in some research (He *et al.* 2004; Modica *et al.* 2001) appears limited. For example, the schema tree which represents a query interface on a Deep Web site is mostly of depth of 3 in (He *et al.* 2004). It will be argued that, in addition, automatic extraction of semantics from Deep Web sites needs to be performed on result pages after a form has been submitted, especially when extracting instances and their corresponding concepts.

Instances are individuals which belong to concepts in an ontology. OntoMiner (Davulcu *et al.* 2003) finds URLs on a partitioned segment of a Web page (e.g., a home page of a hotel) which are linked to instances. From each segment, labels which are concepts and their sub-labels with corresponding values are extracted as instances and encoded in XML. In a case where there is no conspicuous label to cover sub-labels in a Web page, a classifier needs to be developed. The observed precision and recall values for this instance mining were 80% and 91%, respectively based on a manually collected set of instances in (Davulcu *et al.* 2003). On the other hand, a result page after a search, called data page, is a source for extracting instances in DeepMiner (Wu *et al.* 2005). Identifying instances is done by a naïve Bayes classifier, thus frequency is a major tool for finding a concept and its instances on a newly generated result page. The <table> construct in HTML and DOM trees are complimentarily used to identify pairs of concepts and their instance(s) in view of the relative positions of elements. In (Wu *et al.* 2005), precision and recall were also used to measure mining performance. Of course, data to compute these measures were manually obtained, for example, the number of concepts

and instances found in all sampled result pages. At most 41 concept and instance pairs from all result pages represented in seven Dom trees in a certain domain were mined and evaluated in (Wu *et al.* 2005). Instance extraction in both Davulcu *et al.* (2003) and Wu *et al.* (2005) is limited to the Web pages not using the backend database and a small number of instances, consequently, an expanding the *bootstrapped* domain ontology with many instances would be difficult.

In contrast to most ontology learning work, which focuses on Web documents, as in (McDowell *et al.* 2006, Omelayenko 2001, Weber and Buitelaar 2006), the focus in this research is on the Deep Web. Normal ontology learning work depends on linguistic analysis or machine learning methods, which might be difficult for a Deep Web application due to the structure and inaccessibility of the content. Yet, automatically extracting the semantics of Deep Web sources is an important next step for the advancement of E-Commerce.

In this dissertation work, concepts in an ontology, which was automatically generated from the Deep Web in (An *et al.* 2007c), will be used to extract instances for these concepts. The significant difference of the approach, compared with (Davulcu *et al.* 2003; McDowell *et al.* 2006; Wu *et al.* 2005) is that the newly proposed method extracts instances by dynamically probing backend databases of a Deep Web site. This approach is, at least in principle, scalable to very large backend databases.

## 4.3 Enriching a Domain Ontology for the Semantic Deep Web

### 4.3.1 Approach to Instance Extraction

In ontology engineering, concepts and instances are distinguished. In a graph representation of an ontology, concepts are denoted by intermediate nodes and the root while instances are always leaf nodes. (Without instances, a concept may be a leaf.). The previous work (An *et al.* 2007c) and Chapter 3 dealt with the schema level while Chapter 4 deals with the data level, as instances extracted from Deep Web result pages are utilized. While the schema level extraction finds concepts such as 'city,' 'airport code,' etc., the data level extraction results in instances such as 'Newark,' 'EWR,' etc. A data level domain ontology fragment is a set of instances with a corresponding concept whose source is a Deep Web site.

The newly proposed method for extracting instances from the Deep Web is based on developing "robots" (agents, softbots) that send many queries to the same Web site to extract as many data values as possible. Suppose a robot encounters an input field and is not certain what kind of values should be entered. The robot may enter random values or leave the input field empty and then submit the page to the server, to elicit an informative response.

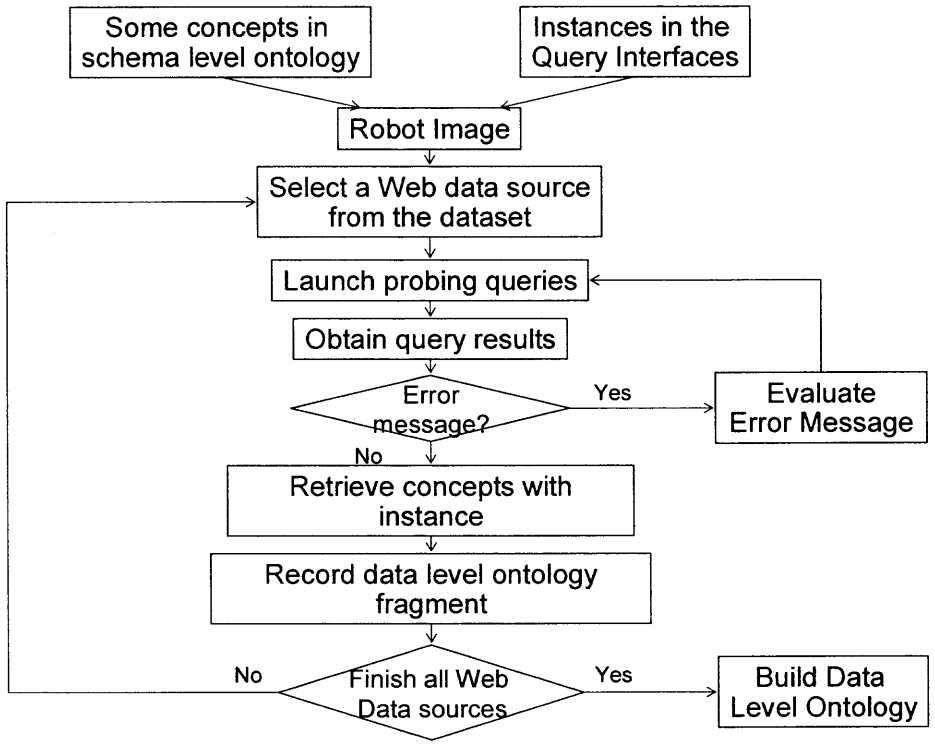



**Figure 4.2** A flow for generating data level ontology fragments.

Figure 4.2 shows the workflow for generating the data level ontology. First, a robot needs a *robot image* to initialize its search for Web data sources. The concept discovery of the robot is guided by a human in its initial stage. In order to help the robot, initial pairs of concepts and their corresponding instances have to be defined, that is called a robot image in this dissertation. Once the robot selects a Web data source, it submits input values into the fields of the query interface. This process is called "generating probing queries." If the input values are not suitable for the form, most Web sites display error messages. The analysis of the error messages often gives useful clues to the robot to guess suitable input values and launch better probing queries. Thus, the queried Web sources may provide rich information about concepts, instances and

semantic relationships. This information is recorded as a data level ontology fragment which is used to refine the prior ontology.

Consider a Web site with a dynamic query interface, such as a flight reservation system. Prior to launching probing queries, the crawling program generates an exhaustive list of suspected candidate airport codes in the range from AAA to ZZZ with the assumption that airport codes consist of three alphabetical characters. The probing program submits the form with a candidate airport code as shown in Figure 4.3, gets a result in HTML and parses it. Next, the program extracts available instances (e.g., airport code, airport name, etc.) from the result Web page given in Figure 4.4. If the candidate airport exists, it will be contained in the HTML page. Otherwise, an error message or a "similar" airport will be returned. This step is referred to as "crawling instances."

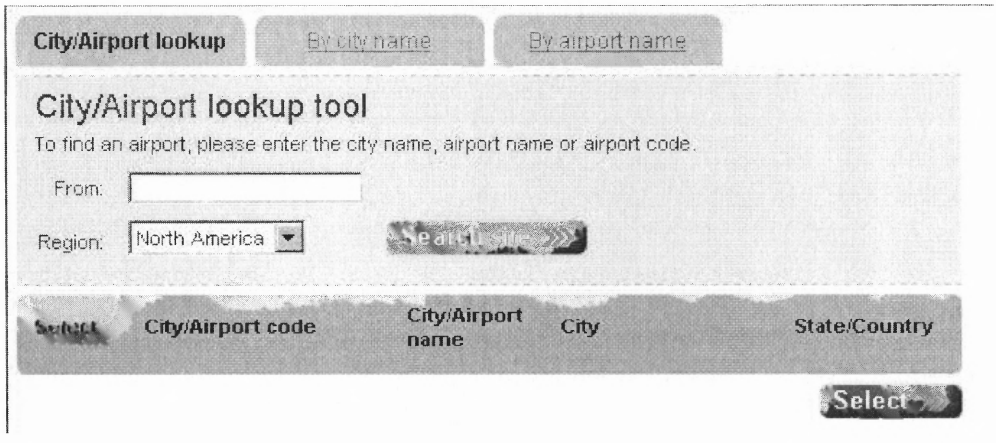



**Figure 4.3** A sample Web site with a dynamic query interface.


**Figure 4.4** A sample web site with results.

In order to extract such instances, a Web crawling system was implemented. The crawling system consists of an applet and JavaScript which run on the research server to communicate with a *probing agency*. The probing agency contains a HTML form which is compatible with the query form of a Deep Web source site and appears in a Web browser. The applet with JavaScript automatically fills in the HTML form of the probing agency and then, the agency connects to the Deep Web source site and submits the form to it. Once the probing agency receives a search result page from the source site, the JavaScript code reads the result page and extracts instances with concepts and

relationships.

As a result of running the probing agency, the system extracted 1090 valid airports, where flights of the airline company may originate. Results are such that the following information has been extracted: 1) An airport code and its name; 2) A city where the airport is located and a country the city is located in; and 3) The neighboring airports that are close to the airport (e.g., LGA, JFK and EWR are neighbors). Table 4.1 shows relationships and corresponding classes with the numbers of instances the system extracted from the Deep Web, using the interface shown in Figure 4.2. Concepts in the first column and their corresponding concepts in the third column are in relationships named in the second column. Instances of the relationships were extracted and the number of the relationships is in the fourth column of the Table 4.1.

**Table 4.1** Instances of Extracted Relationships

| Class Name | Relationship | Class Name | No. of Instances |
|---|---|---|---|
| city | *hasAirport* | Airport | 1090 |
| city | *isCityOf* | country | 997 |
| city | *isCityOf* | province | 362 |
| airport_code | *isAirportCodeOf* | airport | 1090 |
| country | *hasCity* | city | 997 |
| province | *hasCity* | city | 362 |
| country | *hasState* | province | 61 |
| province | *isStateOf* | country | 61 |
| airport | *hasAirportCode* | airport_code | 1090 |
| airport | *isAirportOf* | city | 1090 |
| airport code | *nearBy* | airport code | 102 |
| airport | *nearBy* | airport | 102 |
| airport | *sameAs* | airport_code | 1090 |

In summary, many instances and relationships from a Deep Web site were extracted by the novel probing method which was illustrated in Figure 4.2.

### 4.3.2 Ontology Representation in Web Ontology Language (OWL)

All information of domain ontologies was defined and represented in OWL. In Section 4.3.2, elements of OWL will be introduced which were used in this dissertation and then the automatic creation of the enriched domain ontologies will be described in an implementation view.

**4.3.2.1 Introduction to OWL.** OWL is a Web standard language for ontologies for supporting the Semantic Web, which is endorsed by the W3C (World-Wide Web Consortium 2007). OWL is built on top of the Resource Description Framework (RDF), and helps computers to process Web information (Patel-Schneider 2004).

In OWL, a concept, an instance and a relationship of an ontology are called 'class', 'individual' and 'object property' respectively. An IS-A relationship between two classes is defined as a 'subclass' of a class and consequently, a set of individuals of the subclass is considered as a subset of the set of individuals of its super class. Some constructs of OWL are as follows:

(1) Class, Subclass and Individual: Below, 'city' is described as a class and it is a subclass of a class, 'group' and one of its individuals is 'Newark'

```
<owl:Class rdf:ID= "city">
  <rdfs:subClassOf>
<owl:Class rdf:ID= "group"/>
  </rdfs:subClassOf>
  </owl:Class>
<city rdf:ID= "Newark"/>
```

(2) Object Property: Below, the property 'hasAirport' has a domain of 'city' and a range of 'airport.' The object property 'hasAirport' relates individuals of the 'city' to individuals of the 'airport' such that a city 'Newark' has an airport, 'Newark International Airport.'

```
<city rdf:ID= "Newark">
<hasAirport>
<airport rdf:ID= "Newark_International_Airport">
<isAirportOf rdf:resource= "#Newark"/>
</airport>
</hasAirport>
</city>
```

(3) The built-in Property, *equivalentClass*: An axiom that a class is equivalent to another class indicates that both classes have the same meaning. For example, 'city' is an equivalent class of 'metropolis' such that both classes have precisely the same individuals.

```
<owl:Class rdf:ID= "city">
<rdfs:subClassOf>
<owl:Class rdf:about= "#group"/>
</rdfs:subClassOf>
<owl:equivalentClass>
<owl:Class rdf:ID= "metropolis"/>
</owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about= "#metropolis">
<owl:equivalentClass rdf:resource= "#city"/>
<rdfs:subClassOf>
<owl:Class rdf:about= "#group"/>
</rdfs:subClassOf>
</owl:Class>
```

(4) The built-in Property, *sameAs*: Two individuals can be stated to be the same. In the below example, the individual 'Newark International Airport' is stated to be the same as 'EWR.'

```
<airport rdf:ID= "Newark_International_Airport">
<owl:sameAs>
<airport_code rdf:ID= "EWR"/>
</owl:sameAs>
</airport>
```

(5) The built-in Property, *someValuesFrom*: A particular class may have a restriction on a property that at least one value for that property is of a certain

type. In the below example, the class 'departure city' has a someValuesFrom restriction on the 'hasAirport' property such that some values for the 'hasAirport' property should be an individual of the class 'Airport.' In other words, it is necessary for a class 'departure city' to have at least one airport that is an individual of the class 'Airport.'

```
<owl:Class rdf:ID= "departure_city">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID= "has Airport"/>
</owl:onProperty>
<owl:someValuesFrom rdf:resource= "#airport"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource= "#city"/>
</owl:Class>
```

(6) The built-in Property, *hasValue*: For example, in the below example, a set of individuals whose main office is in Seoul (i.e., 'hasMainOfficeIn' property Seoul ) forms a class, 'Korea_Originated_Airlines.'

```
<city rdf:ID= "Seoul">
<hasAirport>
<airport rdf:ID= "Seoul_Kimpo_International_Airport">
<isAirportOf rdf:resource= "#Seoul"/>
</airport>
</hasAirport>
</city>
<owl:ObjectProperty rdf:ID= "hasMainOfficeIn"/>
<owl:Class rdf:ID= "Airlines_OriginatedIn_Korea">
<rdfs:subClassOf rdf:resource= "#airline"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource= "#hasMainOfficeIn" />
<owl:hasValue rdf:resource= "#Seoul" />
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

There are many other built in OWL properties and restrictions (World-Wide Web Consortium 2004) but the above introduced constrcuts are used for the domain ontologies in this dissertation.

**4.3.2.2 Implementation**. Programs were implemented in order to write a domain

ontology in OWL. In Chapter 4, among eight domain ontologies automatically generated in Chapter 3, the *airfares* domain ontology was enriched with instances and relationships. In all, 1090 instances of airports, 1090 instances of airport codes, 997 instances of cities, 61 instances of provinces and 147 instances of countries with many relationships, as shown in Table 4.1 were extracted from the Deep Web for enrichment.

In order to load this information into the domain ontology, a program was implemented using the Protégé API (Stanford Medical Informatics 2007a) and the Protégé – OWL API (Stanford Medical Informatics 2007b). In detail, classes employed in the loading program are as follows.

(1) RDF:Property: An RDF Resource representing an rdf:Property or an instance of a subclass of rdf:Property such as owl:ObjectProperty and owl:FunctionalProperty. This is used for the *sameAs* property.

(2) OWLObjectProperty: This class implements the RDFProperty interface which provides abstract methods to create new properties, and get or set the domain of the property, get or set the range of the property. In addition, instances can be located in the proper domain and range of each property of a class.

(3) Cls: This Protege Class is used to locate the classes where instances are to be added.

(4) Instance: This Protege Instance (Individual) class is used to write all instances such as airport instances, city instances, etc.

After writing all information in OWL, the enriched domain ontology can be graphically displayed and edited using Protégé – OWL editor (Stanford Center for Biomedical Informatics Research 2007). As shown In Figure 4.5, 1090 airports are listed and for each airport individual, all its object properties are visible. For the airport , 'John F. Kennedy International airport', its property, *hasAirportCode* of 'JFK,' *isAirportOf* 'New York,' and nearBy 'White Plains Westchester County Airport', 'La Guardia International

Airport,' and 'Newark Liberty International Airport' are visible in the Protégé –OWL editor.

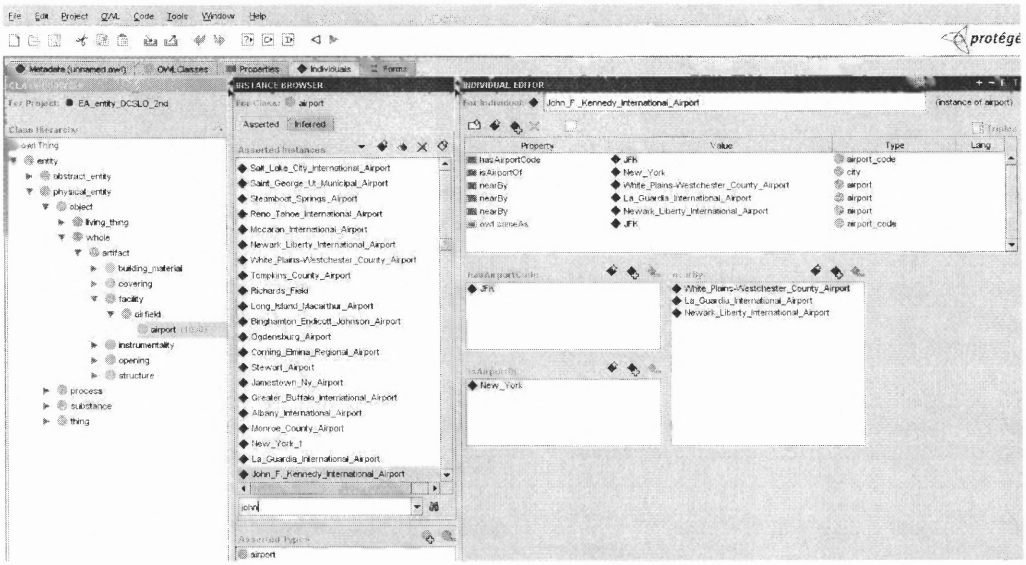


**Figure 4.5** A domain ontology with instances and relationships.

### 4.4 A Web Search with Domain Ontology-based Query Extension

In order to assess the usability of the enriched domain ontology, a domain ontology-based Web search system was implemented. Figure 4.6 illustrates the work flow of the Web search system with the enriched domain ontology.

Suppose that a user typed 'New York' and 'Seoul' into the search input textboxes on the input form (1) to find the air fare between these two cities. The query is parsed (2) into the ontology client (3). The ontology server (4) receives a request from the ontology client (3) to find the semantics of the key words the user typed, and to search for them in the domain ontology (5). Next, the ontology server (4) displays related semantics in the HTML viewer (6), as shown in Figure 4.7, and thus, a user can view semantic choices

and select a few assertions to specify his own interests. Based on the assertions that a user chose, the extended query submitter (7) sends a new query to the Web search engine (8).

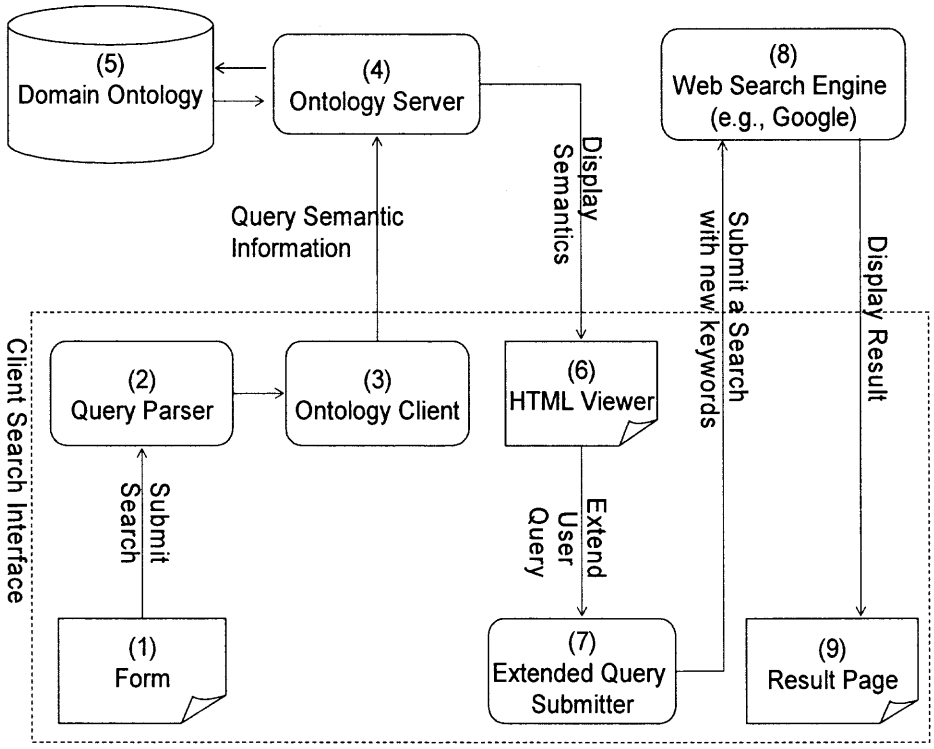



**Figure 4. 6** Work flow of domain ontology-based Web search.

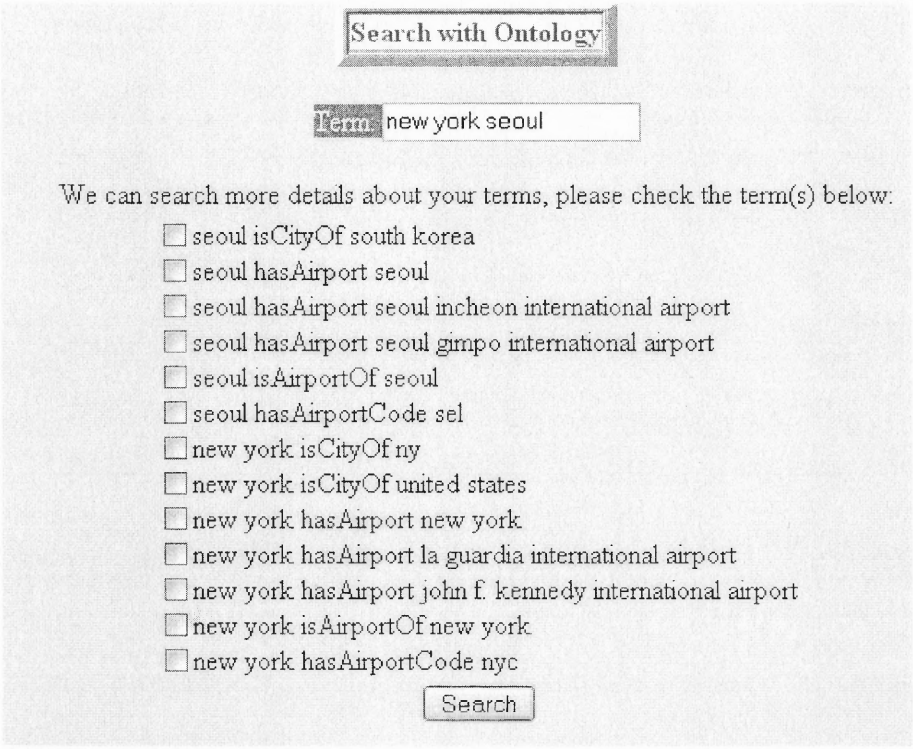



**Figure 4.7** User feedback interface.

For example, if a user clicks on assertions related to airport codes or airports, the extended query submitter (7) will create an extended list of key words (i.e., "New York", NYC, "Seoul", SEL). The assumption that a user inputs only "New York" and "Seoul" as query terms to search for a flight which operates from "New York" to "Seoul" relies on an observation in (Jansen *et al.* 2000). It was reported that a user, on average, enters 2.1 terms for a Web search.

### 4.4.1 Problem Formulation for the Assessment of Ontology-based Web Search

This section attempts to formulate the problem for how to assist users to search Web sites better. Two versions will be presented.

Let $S$ be a string of terms which a user entered for searching. Let $M$ be the set of Websites that the user would consider as matches for his or her search and $W_S$ be all Websites returned by a search engine using $S$. For a perfect Web search engine, $W_S$ should be equal to $M$. In reality, $|W_S| >> |M|$, i.e., while there are a few Websites that the user considers as matches, most Web search engines would return large numbers of Websites. It is hoped by the users that $W_S \supseteq M$ as shown in Figure 4.8 (a) but in reality a Web search engine might miss some Websites in $M$ as depicted in Figure 4.8 (b), i.e., $W_S$ would include only a subset of $M$ denoted as $M_W^S = W_S \cap M$. The set $M - W_S$ contains all misses by the Web search engine using the user's term.

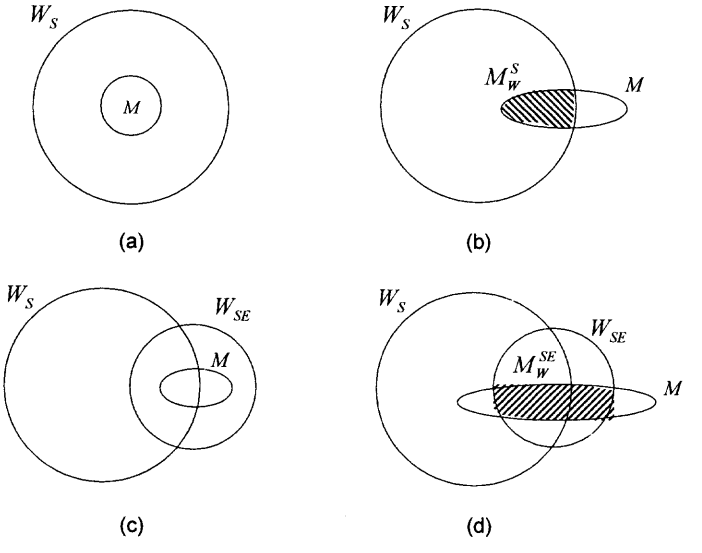



**Figure 4.8** (a) Ideal case for the user's term, (b) typical case for the user's term, (c) ideal case for the extended term, and (d) typical case for the extended term.

Let $W_{SE}$ be all Web sites found by the Web search engine using the extended list of terms which is expected to be logically better than the user's terms. There might exist some Web sites in $W_{SE} - W_S$ and some other Web sites in $W_S - W_{SE}$. It would be ideal if all Web sites, $w$ in $W_S - W_{SE}$ are irrelevant Web sites, i.e., $w \in M^C$ and the set $W_{SE} - W_S$ includes all misses $M - W_S$ as shown in Figure 4.8 (c). A relaxed version is given in Figure 4.8 (d). Let $M_W^{SE}$ be the intersection of $W_{SE}$ and $M$, denoted as $M_W^{SE} = W_{SE} \cap M$. Note that Figure 4.8 (c) is a special ideal case of Figure 4.8 (d).

Consider the following problem, "Find the extended terms so that the Web search results using the extended terms are better than those using the user's terms." This problem can be formalized as follows. Given $S$ (the list of the user's terms), find $SE$ (an extended list of $S$) such that $|W_{SE}| \leq |W_S|$ and $|M_W^{SE}| > |M_W^S|$. First, it should be noted that the inequality $|W_{SE}| \leq |W_S|$ will make browsing faster because the user needs to check a smaller number of returned Web sites for the matches. Next, it should be also noted that the inequality $|M_W^{SE}| > |M_W^S|$ guarantees that the results will be better since the user would find more relevant Web sites, i.e., matches to his or her query. The smaller number of $|W_{SE}|$, together with the larger number of $|M_W^{SE}|$, are *sufficient* conditions for improved Web search results from a user's point of view. Coming up with such an algorithm is infeasible and nearly impossible to be validated. In other words, finding $|M_W^S|$ is practically impossible because no user will be able to validate the entire $|W_S|$ number of Web pages for matches for most searches in case that a large number of Web sites are returned by a search engine.

However, it is feasible for users to review and validate the first one hundred Web

sites for the matches manually. The number of matches in the first one hundred Web sites is without and with extension are denoted as $|(M_W^S)_{100}|$ and $|(M_W^{SE})_{100}|$, respectively. Then the following version of the problem formalization becomes feasible. Given $S$ (the listof the user's terms), find $SE$ (an extended list of $S$) such that $|W_{SE}| \leq |W_S|$ and $|(M_W^{SE})_{100}| > |(M_W^S)_{100}|$.

### 4.4.2 Algorithm

Let a domain ontology $D$ be defined as $<C_D, CI_D, P_D, PI_D>$ where $C_D$ is the set of concepts in the domain ontology, $CI_D$ is the set of individuals of $C_D$, $P_D$ is the set of object properties in the domain ontology, and $PI_D$ is the set of instances of object properties. In this structure, for example, New York *isCityOf* JFK belongs to $PI_D$.

The algorithm for extending a set of query terms that a user inputs, with the help of the domain ontology, is as follows.

**Algorithm 4.1:** Extending User Query Terms

**Input:** $S$: a string with user terms for searching, where $S = \{t_i\}, i = 1, 2, ..., k$ and $t_i$ is a term.

D = $<C_D, CI_D, P_D, PI_D>$ Domain ontology

**Output:** $SE$: an extended string of $S$.

**Steps:**

(1) Compose $S$ into a set of candidate instances, $L = \{prase_{i=1,2,..}\}$, $i = 1, 2, ..., k$ in a lexical order

(2) Initialize a set of instances, $I_{Temp}$, a set of instances of properties, $IP_{Temp}$ and a set of properties, $P_{Temp}$ as empty sets.

(3) **for each** candidate instance $prase_i$ **do** {

**if** $phrase_i \in CI_D$ **then**

$I_{Temp} = I_{Temp} \cup phrase_i$ ; }

(4)**for each** element $k$ in $I_{Temp}$ **do** {

$P_{Temp} = P_{Temp} \cup Get\ Set\ of\ Properties\ (k)$;

**for each** element $m$ in $P_{Temp}$ **do**

$IP_{Temp} = IP_{Temp} \cup Get\ Set\ of\ Instances\ of\ Property\ (k, m)$; }

(5) Display a set of $IP_T$ to a user, to select from

**if** an element $l$ in $IP_T$ is selected **then**

Extract an involved instance from $l$ ;

// for example, "John F. Kennedy International Airport" is extracted from

// "New York *hasAirport* John F. Kennedy International Airport"

Extend the original search string to *SE* by appending the instance;

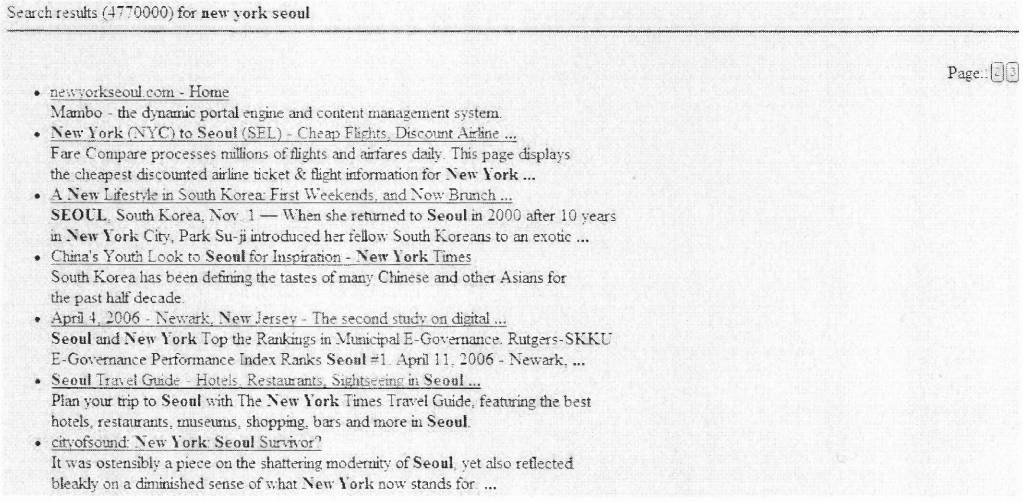
**return** *SE*; // SE can now be plugged into a search engine such as Google.

The algorithm processes a user query string and separates it into phrases (i.e., Step 1). For example, for the string, "Where is New York," the set of the phrases is {"where," "where is," "where is new," "where is new york," "is," "is new," "is new york," "new," "new york" and "york."} Next, if a phrase matches the name of an instance of the domain ontology (i.e., Step 3), extract the properties of the instance by using the Protégé API (i.e., Step 4). For example, an instance "New York" has object properties such as *isCityOf*, *hasAirport*, *hasAirportCode*, etc. and the instances of these object properties are New York, John F. Kennedy International Airport, and NYC, respectively (i.e., Step 5). These are included into the original query terms a user typed in, and the extended query is submitted to the search engine.

### 4.4.3 Experimental Assessment

The success of Algorithm 4.1 can be assessed in two ways: amount and relevance. The first condition for the successful scenario is $|W_{SE}| \leq |W_S|$. The Web site list displayed to the user should be much smaller when the extended list of terms is used, compared to when only the user's terms were used. The relevance condition is $|M_W^S| \leq |M_W^{SE}|$. In the ideal case, i.e., $M \subset W_{SE}$ as shown in Figure 4.7(c), only the first condition is sufficient. Unfortunately, $M$ is not easily accessible and thus efficiency of the newly proposed approach will be supported by a case study with experimental results.

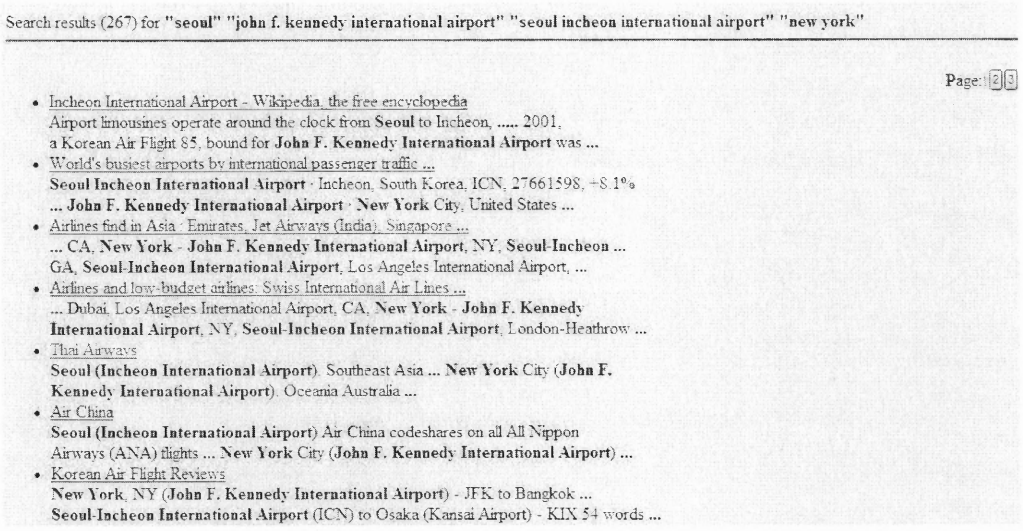
Using Google with the user terms $S$ = {'New York', 'Seoul'} resulted in a display containing only a few flight reservation Web sites. Then the system extended the query string to $SE$ = {'New York', 'John F. Kennedy International Airport', 'Seoul', 'Seoul Incheon International Airport'} with respect to the user selections of semantics in Figure 4.7.


Search results (4770000) for **new york seoul**

Page: 1 2

- newyorkseoul.com - Home
  Mambo - the dynamic portal engine and content management system.
- **New York** (NYC) to **Seoul** (SEL) - Cheap Flights, Discount Airline ...
  Fare Compare processes millions of flights and airfares daily. This page displays the cheapest discounted airline ticket & flight information for **New York** ...
- A **New** Lifestyle in South Korea: First Weekends, and Now Brunch ...
  **SEOUL**, South Korea, Nov. 1 — When she returned to **Seoul** in 2000 after 10 years in **New York** City, Park Su-ji introduced her fellow South Koreans to an exotic ...
- China's Youth Look to **Seoul** for Inspiration - **New York** Times
  South Korea has been defining the tastes of many Chinese and other Asians for the past half decade.
- April 4, 2006 - Newark, New Jersey - The second study on digital ...
  **Seoul** and **New York** Top the Rankings in Municipal E-Governance. Rutgers-SKKU E-Governance Performance Index Ranks **Seoul** #1. April 11, 2006 - Newark, ...
- **Seoul** Travel Guide - Hotels, Restaurants, Sightseeing in **Seoul** ...
  Plan your trip to **Seoul** with The **New York** Times Travel Guide, featuring the best hotels, restaurants, museums, shopping, bars and more in **Seoul**.
- cityofsound: **New York**: **Seoul** Survivor?
  It was ostensibly a piece on the shattering modernity of **Seoul**, yet also reflected bleakly on a diminished sense of what **New York** now stands for ...

**Figure 4.8** Search result page before using Algorithm 4.1.


Search results (267) for "seoul" "john f. kennedy international airport" "seoul incheon international airport" "new york"

Page: 2 3

- Incheon International Airport - Wikipedia, the free encyclopedia
  Airport limousines operate around the clock from **Seoul** to Incheon, ..... 2001, a Korean Air Flight 85, bound for **John F. Kennedy International Airport** was ...
- World's busiest airports by international passenger traffic ...
  **Seoul Incheon International Airport** · Incheon, South Korea, ICN, 27661598, +8.1% ... **John F. Kennedy International Airport** · **New York** City, United States ...
- Airlines find in Asia : Emirates, Jet Airways (India), Singapore ...
  ... CA, **New York** - **John F. Kennedy International Airport**, NY, **Seoul-Incheon** ... GA, **Seoul-Incheon International Airport**, Los Angeles International Airport, ...
- Airlines and low-budget airlines: Swiss International Air Lines ...
  ... Dubai, Los Angeles International Airport, CA, **New York** - **John F. Kennedy International Airport**, NY, **Seoul-Incheon International Airport**, London-Heathrow ...
- Thai Airways
  **Seoul (Incheon International Airport)**, Southeast Asia ... **New York** City (**John F. Kennedy International Airport**), Oceania Australia ...
- Air China
  **Seoul (Incheon International Airport)** Air China codeshares on all All Nippon Airways (ANA) flights ... **New York** City (**John F. Kennedy International Airport**) ...
- Korean Air Flight Reviews
  **New York**, NY (**John F. Kennedy International Airport**) - JFK to Bangkok ... **Seoul-Incheon International Airport** (ICN) to Osaka (Kansai Airport) - KIX 54 words ...

**Figure 4.10** Search result page after using Algorithm 4.1.

In this case, $|W_{SE}| = 267 < |W_S| = 4{,}770{,}000$ which clearly satisfies the first condition. Next, finding $|M_W^S|$ is literally impossible because no user will validate the entire $|W_S|$ number of Web pages for a match. Thus, the first one hundred Web sites were manually reviewed and the relevant web pages, denoted as $|(M_W^S)_{100}|$, in the first one hundred Web sites were counted. With the initial query string, 1 Web sites in the first seven result pages were flight reservation related as shown in Figure 4.9, while five Web sites from the first 7 result pages are sites relevant to flight reservations as shown in Figure 4.10. Unfortunately, $|(M_W{}^S)_{100}| \leq |(M_W{}^{SE})_{100}|$ does not imply $|M_W{}^S| \leq |M_W{}^{SE}|$. It is impossible to validate $|M_W{}^S| \leq |M_W{}^{SE}|$ either theoretically or experimentally. However, the inequality $|(M_W{}^S)_{100}| \leq |(M_W{}^{SE})_{100}|$ is more appealing than $|M_W{}^S| \leq |M_W{}^{SE}|$ because most users would review the first hundred returned Web pages. In other words, no one would review the entire list of returned Web pages spending hours of time and effort.

Table 4.2 lists the case study's results. For example, when a set of user query terms {"New York" "Seoul"} was submitted to the search engine, 4,770,000 was the number of search results while only 267 search results were returned when a set of extend query terms {"New York" "John F. Kennedy International Airport" "Seoul" "Seoul Incheon International Airport"} was submitted. This means, human and search engine processing of the initial user query took longer time than for the extended query terms. With respect to relevance, the initial user query returned nine relevant Web sites in one hundred result pages, while the extended query terms returned forty five relevant Web sites in one hundred result pages for the topic flight reservation. In total, for the five cases, the number of search results was much smaller, namely $\sum_{i=1}^{5}|W_S| = 674{,}770{,}000$ vs. $\sum_{i=1}^{5}|W_{SE}| = 91{,}586$. The relevance was much closer namely $\sum_{i=1}^{5}\left|(M_W^S)_{100}\right| = 21$ vs. $\sum_{i=1}^{5}\left|(M_W^{SE})_{100}\right| = 71$. In this study, it was concluded that ontology-supported search returned more specific results to the user.

Preliminary experiments suggest that Web search results returned by a general purpose search engine such as Google were improved by the newly proposed approach. More sites relevant to a user's needs could be located. The fact $|(M_W{}^S)_{100}| \leq |(M_W{}^{SE})_{100}|$ justifies the claim of 'more relevant sites' and $|(M_W{}^S)_{100}| \leq |(M_W{}^{SE})_{100}|$ together with $|W_{SE}| \leq |W_S|$ supports the claim of 'faster' when $|W_S|$ and $|W_{SE}|$ are so small that users would review all returned Web sites. Users would need to examine fewer results and would find more relevant Websites early on in the result pages.

**Table 4.2** Case Study Results

| $S$ | $SE$ | $\lvert W_S \rvert$ | $\lvert W_{SE} \rvert$ | $\lvert (M_W{}^S)_{100} \rvert$ | $\lvert (M_W{}^{SE})_{100} \rvert$ |
|---|---|---|---|---|---|
| "New York" "Seoul" | "New York" "John F. Kennedy International Airport" "Seoul" "Seoul Incheon International Airport" | 4,770,000 | 267 | 9 | 45 |
| "New York" "London" | "New York" "John F. Kennedy International Airport" "London" "London Heathrow International" | 247000000 | 236 | 3 | 3 |
| "New York" "Paris" | "New York" "John F. Kennedy International Airport" "Paris" "Paris" | 186000000 | 66100 | 4 | 8 |
| "New York" "Miami" | "New York" "John F. Kennedy International Airport" "Miami" "Miami International Airport" | 119,000,000 | 24,600 | 4 | 4 |
| "New York" "Tokyo" | "New York" "John F. Kennedy International Airport" "Tokyo" "Narita International Airport" | 118,000,000 | 503 | 1 | 8 |

## 4.5 Limitations

In an additional experiment, it is observed that up to some number of query terms, the number of relevant sites can be increased. However, after that, the number goes down dramatically, because of a set of too specific, long query terms. It appears that there exists a point up to which a query string should be extended or, similarly, how many semantic choices should be selected. Figure 4.11 shows the correlation between the number of relevant Websites and the number of query terms. Hereby, dynamically adjusting the length of an extended query to guarantee a good result brings another research issue.

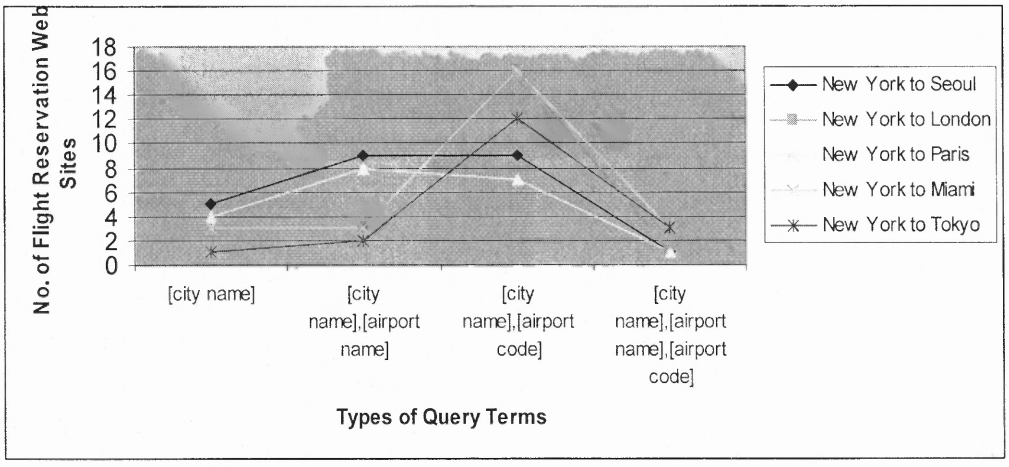



**Figure 4.11** The search results along with different types of query terms.

Naturally, the cases used for assessing the performance of domain ontology-based Web search are limited in this dissertation. For example, some users prefer cheap flights, while other users prefer direct flights or want to fly only with certain airlines. However, the main purpose of implementing the Web search system is to show improvements due to the use of instances in a domain ontology. Research issues when locating search results for a specific Web user interest include how to understand the context of the query, how to give proper responses (Singh 2002), how to best processes queries to return relevant pages (Singh 2002), how to create dynamic pages that need computation on the fly, how to support simultaneous searches (Ghanem and Aref 2004), etc. All these are emerging issues for research on searching the Deep Web.

While a large number of instances was extracted from the structural database of a specific Deep Web site, there are many databases of E-Commerce Web sites which a crawling program cannot access, due to access restrictions. Session tracking by a Web site is a known barrier for a Web crawling program, as a site may use cookies to trace

interactive progress of a client (Liddle *et al.* 2002). In addition, JavaScript may arbitrarily change a data representation which is independent of the visual interface shown to a client. This is another barrier against automatic analysis programs (Liddle *et al.* 2002). In fact, while the crawling program was tested, several problems were detected in reading HTML results from a number of Web sites. A message, "Please activate scripting" means that the site detects a Web browser of a client and its script option before it answers a user query. Thus, a crawling program needs a facility of a kind of Web browser. On the other hand, a message of "Permission denied" was encountered when the JavaScript program, which runs on a university server, directly read result pages which were displayed on the screen (i.e., client side) but streamed from a company Website. This method of access is called *cross-site scripting.* It causes security breaches, and thus, is prohibited.

Automatic downloading of Web documents from a hidden Website, studied in (Ntoulas *et al.* 2005), is relatively easier than retrieving information from a structured backend database in an interactive fashion. In fact, a crawling program, which failed to read a Deep Web page's HTML content on the fly, could automatically download Web content (http://www.jamia.org/cgi/reprint/M2314v1.pdf) from a textual database such as PubMed, without a security related error message. Considering this paper's research result, namely that ontology instances extracted from the Deep Web contribute greatly to locating relevant Websites, community wide efforts are necessary to extract large numbers of instances from the Deep Web. Only a collective approach can make the Semantic Deep Web a reality.

# CHAPTER 5

# ONTOLOGY EVALUATION: NATURALNESS PERSPECTIVE

## 5.1 Introduction

This chapter deals with aspects of the quality of ontologies (QoO) which is increasingly becoming a research issue on the Semantic Web, specifically the notion of naturalness of ontologies will be focused. The term naturalness introduced by McCray *et al.* (2001) was regarded as the acceptance of the ontology by domain experts. In this dissertation, by extending this idea to regular (non-expert) Web users, naturalness will be refined to be a *measurable* and desirable property of ontologies. Furthermore, by conducting the measurement and comparative analysis of the naturalness on several existing, major ontologes, a mechanical approach is proposed to improve ontology usability on the Semantic Web.[1]

In his original work on ontologies, Gruber (1993) stressed that ontologies are about knowledge sharing. The question must be raised whether existing ontologies are constructed so that they may succeed at this task. In a recent study, Zeng *et al.* (2005) showed that communication through terminologies can be significantly facilitated if words labeling concepts are comprehensible to users. Finding concepts which are likely to be recognized by users is a new trend in ontology engineering, which is different from the traditional approach of building terminologies understandable mainly by experts of a domain. The latter case could cause difficulties in understanding and using ontologies for emerging user communities on the Semantic Web.

---

[1] This chapter's contents were published in (An *et al.* 2006; An *et al.* 2007b).

In recent papers (Lee and Geller 2005; An *et al.* 2006), it has been pointed out that unnatural concepts make it difficult to use an ontology and they contradict the desiderata of an ontology, which include explanatory power for the purpose of sharing information. Specifically, this thesis concentrates on the concepts and concept pairs used in IS-A relationships and semantic relationships in existing ontologies. It should be noted that this dissertation is limited to ontologies with concepts that are labeled in English. In (An *et al.* 2006), it was observed that many labels correspond to English words that can be located in a dictionary. However, this is often not the case for concepts with names consisting of several words. Concept labels for which dictionary lookup fails to find an entry and for which the compositional meaning is difficult (if not impossible) to derive from its components, in a consistent manner for different participants in an act of communication, were also found. Thus, a sender of such a label might assume a different meaning than the one understood by a receiver. For example, it is hard to judge what the exact meaning is of Partially Intangible Individual (Cycorp 2005).

This thesis focuses on an ontology's role in knowledge sharing supported by an explicit specification of a conceptualization. The key idea of naturalness is based on this practical definition (An *et al.* 2006). Some researchers (Staab and Maedche 2000) have made efforts in explicating the meaning of semantic relationships by using axioms. However, this declarative knowledge with universal truths about concepts cannot provide answers for all the forms of knowledge inquiries (Mizoguchi 1996). It is widely assumed that ontologies represent information in a form that is at least similar to how human knowledge is represented (Smith 1982). To many researchers, an ontology concept is a meaningless label unless it is given a definition. However, any definition itself will

contain logical symbols and other labels. Logical symbols do not cause a problem because they are domain independent. However, how can the defining labels themselves be defined? This leads to an infinite regression or circular definitions. Thus, it is assumed that at some level labels have to be understandable by being known to the recipient (program or human). As there are labels that are better known, which are called "more natural," and labels that are less well known to humans, the more natural labels are preferred to be used even for programs. It is noted that meaning and naturalness of concepts are orthogonal. This will be discussed in more detail in Section 5.2.

Note that the distinction between primitive and defined concepts is not employed in this research. It is easy to give precise definitions in mathematically oriented domains. However in real world applications the number of primitive concepts is often greater than the number of defined concepts. This results in a structure with a large number of concepts for which no definition is attempted, as they are primitive. Thus the distinction is not really helpful for us.

McCray *et al.* (2001) introduced the idea of naturalness as the acceptance of the ontology by domain experts. In this dissertation, this idea is extended to regular (non-expert) users. Thus the acceptance of an ontology by Semantic Web users is desirable. However, it is impossible to consult many users whether they understand ontology concepts and relationships. In all, a mechanical way to measure this naturalness is necessary.

The use of ontologies as part of the Web is desirable, since it could support finding better answers for users' queries. For example, ontologies may supply generalized terms for a user's Web search terms. An answer for a query could be derived by using

specialization and/or generalization relationships between the concepts of an IS-A hierarchy. Finding broader or narrower concepts of a given concept is an important recommended Web search strategy (UC Berkeley 2006). Hereby, the concept pairs in an IS-A relationship should be closely related and it is assumed that they tend to co-occur on Web pages. Under this assumption, which needs further investigation, the naturalness of an IS-A hierarchy may be quantified by co-occurrence of IS-A-connected concepts on Web pages. This idea is discussed in the ontology community (An *et al.* 2007).

According to Kalfoglou and Hu (2006), application ontologies are converging with the Web. Thus the knowledge provided by ontologies should be refined dynamically by the understanding of Web users.

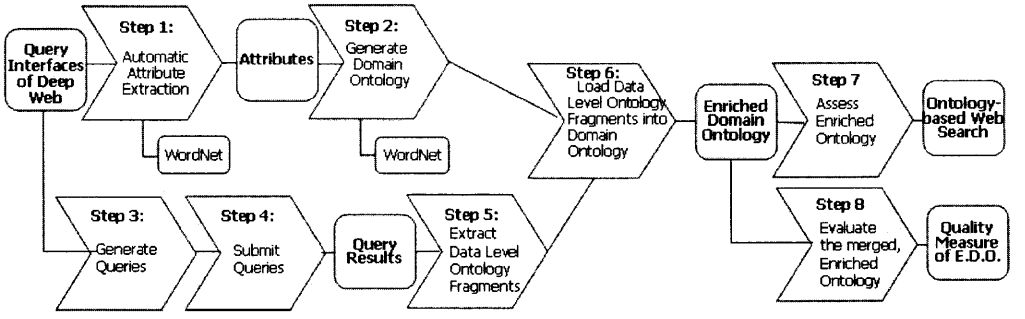



**Figure 5.1** Step 8: evaluating the quality of domain ontologies.

This chapter is about a part of the grand system architecture in Figure 5.1marked colored Step 8. This chapter is organized as follows. Section 5.2 presents a literature review and refines the research questions about the quality of ontologies, especially *naturalness*. A proposed methodology is described in Section 5.3. The results of the study are shown in Section 5.4. Especially, an evaluation of the quality of the automatically generated

ontology with the methodology developed in this dissertation is presented in Section 5.5. Limitations are presented in Section 5.6.

## 5.2 Naturalness Formalization

This section introduces the notion of naturalness as an objective Quality of Ontologies based on their concepts, IS-A relationships and semantic relationships.

### 5.2.1 Naturalness of Concepts

Knowledge engineers use formal or semi-formal languages in order to build ontologies (Colomb 2002). A language provides us with interchangeable words, so the words, which are comprehensible by users, are an important factor when ontologies (Lewis 1983) are evaluated. Naturalness of the concepts used in an ontology is addressed in (An *et al.* 2006), based on the question whether the concept labels are comprehensible to the users.

In (An *et al.* 2006), the following measure of the naturalness of an ontology was proposed. If an ontology consists of (a majority of) labels that can be found in a dictionary lookup, then this ontology is more natural than an ontology for which this is not the case. In order to assign a numeric value to naturalness, it is necessary to know how often a term is used, e.g. by searching a large corpus. In recent years, the Internet has become popular as an ersatz corpus, and, luckily, Google provides us with frequency information also. The *Google#* has been often used in ontology evaluation or matches (An *et al.* 2006; Gligorov *et al.* 2007).

Here, it is claimd that the higher $Google\#(c_i)$, the more natural the concept $c_i$ is. For example, a Google search for DNA finds 118,000,000 hits, while a search for

Deoxyribonucleic Acid finds only 1,750,000 instances. Note that naturalness and meaning are orthogonal. Thus DNA and Deoxyribonucleic Acid have the same meaning. Yet, DNA, commonly used in the popular press nowadays, is a more natural term to most people than Deoxyribonucleic Acid (An *et al.* 2006).Two terms may have identical in meanings, however, they may be of vastly different naturalness. Thus, varicella and chicken pox are identical in meaning. However human subjects (Chun and Geller 2008) consider chicken pox is much more natural.

One purpose of this paper is to formalize the notion of "naturalness." Researchers are often challenged by the existence of a term that is widely used in everyday life, yet no formal definition exists for it. Thus, before Newton, "force" must have been used in everyday language, without users understanding its relation to mass and acceleration. Similarly, the term "naturalness" seems to be well understood in everyday language. A small pilot experiment (Chun and Geller 2008) was performed where humans were asked to judge which term from a pair of terms was more natural to them. This was done for 37 pairs. Not one of the subjects asked for additional clarification, beyond the one given, what "natural" is supposed to mean. Subjects were also in good agreement with each other concerning naturalness. On average, 78% of subjects agreed on naturalness judgments. One goal of this paper is to provide a criterion for naturalness that is independent of human subjects' judgments.

Zheng *et al.* (2005) stressed the importance of consumer-friendly display of medical terms. In their experiment they used medical terms from MedlinePlus and showed that terms more likely to be entered by "end users" as search terms are more likely to be understood by human subjects than less common synonyms of those search

terms. Their paper stresses the importance of (medical) concepts being understandable to non-expert human users. Zheng *et al.*'s term "consumer-friendly" appears to be roughly equivalent to this dissertation's use of the term "natural," except that the term does not assume a reference population ("non-expert users").

Regarding the choice of a proper corpus for finding concept pairs, in (Brewster *et al.* 2002), the Internet and glossaries are reported to be practical resources. That is because concept pairs randomly selected from a domain specific ontology are rarely found in related journals or texts. Practically, the documents on the Internet have been used as alternative corpus to automatically extract hyponyms in (Agirre *et al.* 2000) and examples of concepts (Leacock *et al.* 1998; Mihalcea and Moldovan 1999). Accordingly, in the analysis, the Web, indexed by Google, is adapted as the benchmark corpus for measuring naturalness.

Let's settle on the assumption that most people who make the effort of setting up a Web page have the desire to be understood by other Web users, i.e., they would make their best efforts to use understandable terms. Over 6 billion Web pages indexed by Google support this assumption by the law of large frequency numbers. It can be shown that understandable terms occur more often than obscure terms. Zeng *et al.* (2005) made the assumption that users of a Web site prefer familiar expressions as search terms and thus frequency of used words implies the respective degree of user comprehensibility of the words. In (An *et al.* 2006) the assumption was that a term that is widely used is more likely to be natural.

Let $L = \{l_1, l_2 \cdots, l_n\}$ be a set of labels which appear in an ontology where $l_i$ is a label and $|L| = n$ be the size of $L$ or the total number of labels. Then the naturalness of

concepts for a certain ontology, $O$ is defined as follows:

$$\text{Concept\_ Naturalness}(O) = \frac{\sum_{i=1}^{n} Google\#(l_i)}{|L|} \tag{5.1}$$

### 5.2.2 Naturalness of IS-A Relationships using Frequencies of Concept Pairs

This thesis work uses a similar but more complex approach for naturalness of IS-A relationships. An IS-A relationship is a binary relation which takes two concepts. A notation $X \Rightarrow Y$ is used to denote that X IS-A Y. If an ontology contains the relationship X IS-A Y then this would be considered as a natural statement if many documents that contain both X and Y are not found. If very few documents that contain both X and Y are found then X IS-A Y would be considered as unnatural relationship. An analysis in this dissertation is based on the number of search results for concept pairs returned by Google. For example, let X and Y be two concepts shown in an ontology in which X and Y are in an IS-A relationship. $Google\#(X \wedge Y)$ will be used, called Concept Pair Google Number (CPGN) in this dissertation, indicating the number of co-occurrences of $X$ and $Y$ on the Web pages. CPGN is the number of Web pages found when human query Google for both X and Y in the search. An estimate of this number is reported by Google at the top of every search result.

Let $A$ be a set of IS-A relationships which appear on an ontology where $a_i = P_i \Rightarrow C_i$ is an IS-A relationship $P_i \Rightarrow C_i$ and $|A|$ be the size of $A$ or the number of IS-A relationships. $P_i$ and $C_i$ are concepts such that $P_i$ IS-A $C_i$. Then the naturalness of IS-A relationships for a certain ontology, $O$ is defined as follows:

$$\text{naiveISA\_Naturalness}(O) = \frac{\sum_{i=1}^{|A|} Google\#(P_i \wedge C_i)}{|A|} \tag{5.2}$$

Clearly, an IS-A relationship is asymmetric where $X \Rightarrow Y$ holds but not necessarily $Y \Rightarrow X$. As one of reviewers for this manuscript pointed out, CPGN treat the IS-A relationship as a symmetric relation where both $X \Rightarrow Y$ and $Y \Rightarrow X$ holds or no one can tell the direction by CPGN. This concern is imperative and problematic in building ontologies using co-occurrences of concept pairs (Maedche & Staab 2000), but not for evaluating existing ontologies. It is stressed that our task is *not* finding IS-A relationships by finding co-occurring concepts C and P in the same Web page. Indeed, when finding co-occuring concepts, whether $C \Rightarrow P$ holds, $P \Rightarrow C$ holds, no relationship at all holds, or the relationship between $C$ and $P$ is not an IS-A relationship is can not be known. Rather, it is already known that an IS-A relation between $C$ and $P$ holds in a given direction. Otherwise it would have to be assumed that the designers of the ontology that are being evaluated have made a gross mistake.

Assuming that $C \Rightarrow P$ is correct in that an IS-A relationship between these two concepts really holds, the question is only whether this is a *natural IS-A relationship*. If Web search indicates that there are very few Web pages that contain $C$ and $P$ together, then it may be concluded that there is no strong evidence that any relationship between $C$ and $P$ holds. Therefore, an IS-A relationship is not likely. However, such a relationship was asserted by an ontology designer (who is assumed not to have been grossly negligent). Therefore, it is concluded that the asserted IS-A relationship is correct but is likely not natural in our sense.

One might also wonder why the terms X and Y are allowed to occur anywhere within the document, possibly not connected by an IS-A link, or worse, possibly not connected at all. Wouldn't it be better to look for a sentence frame of the form "X is a Y"? While this approach has intuitive appeal, there are two problems with it. (1) IS-A relationships can be expressed in many different ways, not just by an explicit statement. Thus "animals such as dogs and cats" expresses two IS-A statements. (2) A few experiments querying Google with such sentence frames such as "Xes are Ys" have been done and it is found that they work reasonably well for one-word concepts. However, many concepts are expressed by two or more words, e.g. Amino Acid. For sentence frames involving multi-word X and Y concepts, Google returned small hit counts, and in many cases no hits at all. For example, Google returns no document with a search string, "Amino Acid Sequence is a Molecular Sequence" or "An Amino Acid Sequence is a Molecular Sequence" while it returns 1,070,000 Web documents with a search expression consisting of the two strings "Amino Acid Sequence" and "Molecular Sequence."

*An Initial Experiment*: It is assumed that if $X_i$ and $Y_j$ are indeed in an IS-A relationship then they are closely related and tend to co-occur. In order to support this assumption, one simple experiment was conducted in which the concept pairs in IS-A relationships and non-relevant concept pairs were compared with respect to CPGN. For each concept pair $(X_i, Y_j)$, a non-relevant concept pair was generated as $(X_i, \mathrm{Z_k})$ or $(\mathrm{Z_k}, Y_j)$ where $Z_k$ is a randomly selected concept. A group of these non-relevant concept pairs is the control group to verify the assumption that the naturalness of an IS-A relationship can be approximated by a Google search. The result of the comparison shows that submitting concept pairs with IS-A relationships to Google results in conspicuously

higher numbers of frequency results than for non-IS-A-related concept pairs (see Appendix). So, it can be concluded that CPGN can be used as one measurement to distinguish pairs connected by IS-A relationships from random pairs. Thus, frequencies of co-occurrences by Google search are applied to the measurement of naturalness. The lists of concept pairs used for this experiment are in the Appendix. Therefore, low CPGN values indicate that the existence of an IS-A relationship is unlikely. Thus, frequencies of co-occurrences by Google search to the measurement of naturalness are applied.

### 5.2.3 Naturalness of IS-A Relationships using Rule Mining

A co-occurrence frequency based rule mining technique has been used to build some ontological relationships in (Maedche and Staab 2000). In this thesis's approach, the support and the confidence values are used to determine the degree of association between the concepts of a pair.

The rules for determining the support and the confidence for each concept pair are shown below. Once again, it is assumed that if $X_i$ and $Y_j$ are indeed in an IS-A relationship then they are closely related and tend to co-occur. If the absolute number of co-occurrences is low, this would not be useful. If $X_i$ occurs many times without $Y_j$, this is also not useful. For the successful assessment, there should be many co-occurrence instances, and $Y_j$ needs to appear reliably most of the time when $X_i$ appears. These two parameters are measured in rule mining by support and confidence (Maedche and Staab 2000).

Confidence: The confidence of the $X \Rightarrow Y$ relationship, denoted by $Confidence(X \Rightarrow Y)$, presents the percentage of the obtained CPGN relative to $Google\#(X)$. $Google\#(X)$ indicates the frequency result when only one word "X" is passed to Google. Therefore, the $Confidence(X \Rightarrow Y)$ is defined as

$$Confidence(X \Rightarrow Y) = \frac{Google\#(X \wedge Y)}{Google\#(X)} \tag{5.3}$$

The support of the $X \Rightarrow Y$ relationship, denoted by $Support(X \Rightarrow Y)$, presents the importance of a concept pair frequency result among all the concept pair frequency results. It can be obtained by several steps as shown below. Let $Total\_Google\#(O_x)$ be the summation of $Google\#(X_i \wedge Y_j)$ of all the selected concept pairs in an ontology, called $O_x$. It can be obtained by

$$Total\_Google\#(O_x) = \sum_{(X_i IS-A Y_j) in O_x} Google\#(X_i \wedge Y_j) \tag{5.4}$$

where $X_i$ and $Y_j$ are the concepts in $O_x$ such that the $X_i \Rightarrow Y_j$ relationship holds, and $X_i \neq Y_j$. The support of the $X \Rightarrow Y$ relationship can be obtained by dividing the summation of all frequency results in $k$ ontologies, $\{O_1, O_2, \cdots, O_k\}$.

$$Compound\_Google\# = \sum_{k=1}^{Last\ Ontology} Total\_Google\#(O_k) \tag{5.5}$$

$$Support(X \Rightarrow Y) = \frac{Google\#(X \wedge Y)}{Compound\_Google\#} \tag{5.6}$$

Note that "all" means "all sampled". The ontologies that are used in this chapter are too large to process all pairs of concepts exhaustively. Typically in our experiments in later sections, when two ontoglogies $O_\alpha$ and $O_\beta$ are compared.

Following the results of the *initial experiment* described above (and in the Appendix), it is assumed that the naturalness of an ontology is correlated with the $Confidence(X \Rightarrow Y)$ of all concept pairs. However, a problem arises if the analysis is performed directly on the obtained confidence values, since the confidence only provides the relation between the concepts of a pair without considering all other concept pairs. For example, $Confidence(X_1 \Rightarrow Y_1) = 100\%$ can be obtained from a concept pair for $X_1 \Rightarrow Y_1$ for which $Google\#(X_1) = 2$, $Google\#(Y_1) = 2000$, and $Google\#(X_1 \wedge Y_1) = 2$. On the other hand, $Confidence(X_2 \Rightarrow Y_2) = 90\%$ can be obtained from another concept pair for $X_2 \Rightarrow Y_2$ in which $Google\#(X_2) = 1000$, $Google\#(Y_2) = 2000$, and $Google\#(X_1 \wedge Y_1) = 900$. If the confidence results are concerned only, the concept pair $X_1 \Rightarrow Y_1$ is better than $X_2 \Rightarrow Y_2$. However, this is not correct. Therefore, in order to correctly analyze the QoO, it is necessary to consider the issue how much one can rely on the obtained confidence results. The degree of natural association (DoNA), $DoNA(X \Rightarrow Y)$, of the concept pair was developed to solve the above problem, and is defined as

$$DoNA(X \Rightarrow Y) = Support(X \Rightarrow Y) \times Confidence(X \Rightarrow Y) \tag{5.7}$$

Then the second measure of naturalness of IS-A relationships for a certain ontology $O$ will be called Rulemining-based Naturalness of IS-A Relationships, and is defined as follows:

$$\text{RulemeningISA\_Naturalness}(O) = \frac{\sum_{i=1}^{|A|} DoNA(a_i)}{|A|} \tag{5.8}$$

Note that, once one obtains the DoNA, the analysis of the natural association of concept pairs in ontologies will be conducted by the statistic methods (Section 5.3.2).

### 5.2.4 Naturalness of Concept Pairs Connected by Semantic Relationships

A similar approach as in section 2.2 is used for pairs of concepts connected by semantic relationships. An ontology contains not only hierarchical classifications but also other relationships which enrich data semantics. In (Seta *et al.* 1997), taxonomy and axioms are regarded as the major components of an ontology. The taxonomy consists of IS-A relationships and axioms define rules, constraints and relationships among concepts. There are various ways to represent data semantics. Different domain exerts may perceive the same domain in slightly different ways (Mizoguchi and Ikeda 1996). If an ontology on the same topic is designed by different knowledge engineers, there will be natural variations between the results, as each one is bringing his own perspective to the task.

Generally, the knowledge engineers need to explore related concepts (Brewster *et al.* 2003; Hearst 1992). In addition, frequencies of co-occurrences of the concepts may be used to compute a "semantic distance" between the non-taxonomic concepts (Brewster *et al.* 2003). Thus, in this paper, naturalness of concept pairs connected by semantic relationships is measured by CPGN described in the above section. $Google\#(X_i \wedge Y_j)$ is obtained where $X_i$ and $Y_i$ are semantically related. It is assumed that if the sampled concepts which are defined as semantically related have a high frequency in the

benchmark corpus, Google, the semantic relationship of $X_i$ and $Y_i$ is natural. That is, the higher the $Google\#(X_i \wedge Y_j)$ is, the more natural the semantic relationship is.

Let $S$ be a set of semantic relationships which appear in an ontology where $s_i$ is a semantic relationship $X \Leftrightarrow Y$ and $|S|$ is the size of $S$, i.e. the number of semantic relationships. Then the naturalness of semantic relationships for a certain ontology, $O$ is defined in eqn (9) which is analog to eqn (2):

$$\text{semantic\_Naturalness}(O) = \frac{\sum_{i=1}^{|S|} Google\#(s_i)}{|S|} \tag{5.9}$$

The definition of the semantic relationships is based on the ontologies subject to our analysis. Let $R$ be the set of all binary relationships of all ontologies in the domain. The set of semantic relationships, $S$ = *{IS-A – inverseIS-A}*. Note that $R$ is provided by ontologies in two different ways; while some ontologies provide explicit table listing all binary relationships, others provide on a list of functions. The set $S$ for the UMLS is given in Appendix and the set S for the WordNet and the OpenCyc is described in Section 5.3.1.

## 5.3 Methodology

### 5.3.1 Data Sources

Three major ontologies, WordNet, UMLS and OpenCyc are investigated in this dissertation.

**5.3.1.1 WordNet**. WordNet is a large scale lexical reference system for the purpose of natural language processing. It contains 117,798 nouns in the version 3.0 and the average

number of children for each noun is 1.027. Therefore, it can serve 117,798*1.207 parent-child pairs (Princeton University, 2006). The WordNet 2.0 which is a version for Windows was used for the experiment.

In WordNet, there are other relationships, such as *substance_meronym*, *substance_holonym*, *part_meronym*, *part_holonym* and *entailment*, *entailed_by*, *derived* and *cause* (Didion, 2003). Generally, holonymy is the part-whole relationship such that X is in Y or X is a part of Y. Meronymy is holonymy's inverse. For example, brain cell and brain are in a part_holonym relationship while law of gravitation and gravitational constant are in a *part_meronym* relationship. These relationships and their corresponding concepts are the subjects of this analysis.

**5.3.1.2 OpenCyc**. OpenCyc is the open source version of CYC, which is for general knowledge processing. The name CYC is based on the word EnCYClopedia, and CYC was built with the intention of providing encyclopedic knowledge of concepts and reasoning rules. OpenCyc had 47,000 concepts in the initial release, and 300,000 in the release 0.9 (Cycorp, 2005) which was used for this experiment.

Individuals, also known as instances, represent objects in the domain of interests. Classes provide an abstraction mechanism for grouping resources with similar characteristics (Bechhofer 2004). Classes are organized into a superclass-subclass hierarchy, which can usually be shown as a Directed Acyclic Graph structure. Subclasses specialize their superclasses; superclasses generalize their subclasses. A class maybe associated with a set of individuals, which are the leaves in the hierarchical structure and can not have any sub-individuals. For example in Figure 5.2 (a), consider the classes

"Canine Animal" and "Dog" and the individual "German Shepherd Dog." Dog is a subclass of Canine Animal; Canine Animal is the superclass of Dog. It can be said that, 'All members of the class Dog are members of the class Canine Animal'. German Shepherd Dog is one of the instances of Dog, which can not have any members.

In OpenCyc, *predicate* and *function-denotational* are kinds of relations (Cycorp 2006) which are constraints and relations between concepts. For example, a predicate, *performedBy* which is preceded by a concept, *PettingAnAnimal* limits a following concept to be a Person. So, only if individuals conforming to the above concepts are in a sentence like *PettingAnAnimal*, *performedBy Person* is the sentence semantically well-formed (Cycorp 2002). In the analysis, predicate relations are only considered and function-denotational relations are discarded since functions are used to generate new concepts.

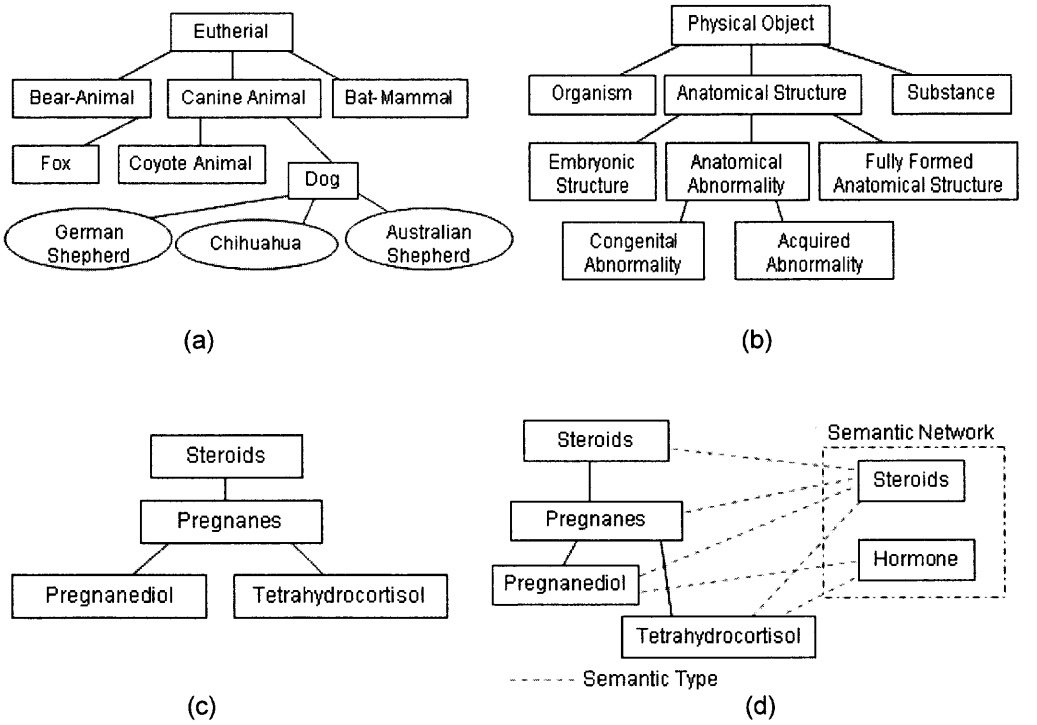



**Figure 5.2** (a) Partial OpenCyc diagram (b) a part of the Semantic Network (c) an example of IS-A relationships in the Metathesaurus (d) an example of assignments of semantic types to concepts in the Metathesaurus of the UMLS.

**5.3.1.3 UMLS (Unified Modeling Language).** The UMLS is a large-scale knowledge base used in Medical Informatics. The UMLS consists of several parts of which we are interested in the Metathesaurus and the Semantic Network with its semantic relationships. The Metathesaurus had 1,436,586 concepts and 7.2 million terms in the version of 2007AB (U.S. National Library of Medicine 2007b) and the Semantic Network has 135 terms, with 612 relationships. There are 135 IS-A relationships between semantic types, 54 IS-A relationships between relationships of the Semantic Network, and 423 non-IS-A relationships of the Semantic Network (Lister Hill National Center 2006). As the Semantic Network is a tree with two roots, Entity and Event, there are 133 IS-A links;

Entity has 99 descendants and Event has 34 descendants. The version of 2005AB was used in this dissertation.

The UMLS Semantic Network contains semantic types and is used to provide the consistency for the Metathesaurus concepts. Each Metathesaurus concept is assigned at least one semantic type. The relationships between the semantic types provide the structure of the Semantic Network which in turn provides important implications for interpreting the meaning of the Metathesaurus concepts. When assigning semantic types to Metathesaurus concepts, the most specific semantic type in the structure is used (U. S. National Library of Medicine 2006). Figure 5.2 (b), (c), and (d) show examples of IS-A relationships of the Semantic Network, the Metathesaurus and semantic type assignments of the Metathesaurus, respectively.

In the UMLS Metathesaurus, concepts are closely connected by certain relationships since they have common properties even in their definitions. Most of relationships come from the same sources called intra-source relationships (U.S. National Library of Medicine 2007a). For example, there are over 25 million of relationship instances between two concepts in the major relationships files, "mrrel.rrf" and the relationships such as *inverse_isa*, *has_permuted_term*, *permuted_term_of* and *sib_in_isa* are ignored in this analysis. The general labels of relationships can be found in the UMLS file, "mrdoc.rrf." On the other hand, the UMLS Semantic Network contains the relationships between pairs of semantic types. 423 relationship instances can be found such as, *treats*, *prevents*, *complicates* and so on.

Even though the UMLS is strictly speaking not an ontology, it is close enough for our purposes to treat it in the same way as the other terminologies. In summary, the

following seven components are used in this dissertation: **(1)** WordNet, **(2)** UMLS Semantic Network, **(3)** UMLS Metathesaurus, **(4)** sets of pairs $(p,r)$ where $p$ is a concept of the UMLS Metathesaurus and $r$ is a Semantic type of the UMLS Semantic Network and $r$ is assigned to $p$ in the UMLS, **(5)** OpenCyc Class, **(6)** OpenCyc Individual, and **(7)** OpenCyc (complete).

The flow of this research is briefly described in Figure 5.3. Since the necessary data comes from different sources, several programs were implemented to gather the data.

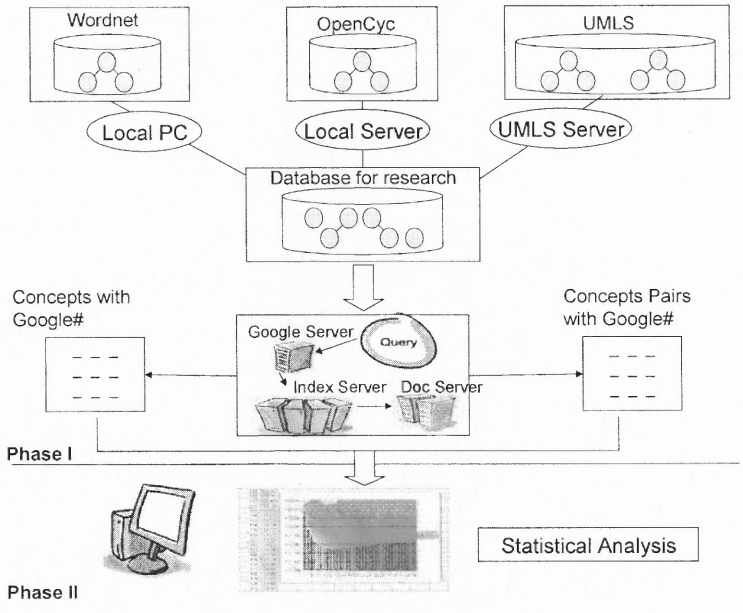



**Figure 5.3** The flowchart of our analysis (An *et al.* 2006).

### 5.3.2 Phase I: Extract Data

To retrieve words (concepts) and their relationship from WordNet, JWNL (Open Source Technology Group 2006b), an *API* (application programming interface) for accessing WordNet-style relational dictionaries, is used to access the WordNet database (Princeton University 2005). For OpenCYC, the database server (Cycorp 2005) is run on a local host

and the OpenCYC API (Open Source Technology Group 2006a) was used to access the database. The UMLS offers a SQL query capability in XML format to directly get records from its database, hosted at umlsks.nlm.nih.gov, by using Java Remote Method Invocation. The table "mrrel" in the UMLS database is used to store all IS-A records for the Metathesaurus. After all records are stored in a local database, relationships between Metathesaurus and Semantic Types can then be built. The UMLS API (U. S. National Library of Medicine Aug. 2005) is used to retrieve the assignments of semantic types for each Metathesaurus term. To retrieve relationship between semantic types, the "srstr" table in the UMLS database has all the necessary records. In a similar way, all sampled non IS-A relationships were extracted from the UMLS Semantic Network and UMLS Metathesaurus. Note that the Metathesaurus of the UMLS, which contains IS-A relationships was used in its entirety. But some semantic relationships mentioned previously were removed in the analysis of semantic relationships. For semantic types, the Semantic Network was also used in its entirety, as it is of moderate size.

**5.3.2.1 Approximating the Naturalness of Concepts for an Ontology**. Algorithm 1 for extracting concepts depends on APIs and the structures of each ontology and can be described as follows.

| **Algorithm 5.1: Approximating the naturalness of concepts for an ontology** |
|---|
| **Input:** a certain ontology $O$, represented as a set of concepts (lables) $L = \{l_1, l_2 \cdots, l_n\}$ |
| **Output:** $L' = \{l_1', l_2' \cdots, l_m'\}$ where $l_i' \in L$ and $l_i' \in L'$ and $m << n$ where $L'$ is a randomly selected sample set of labels and |

$$\text{Concept\_Naturalness}(O) = \frac{\sum_{i=1}^{m} Google\#(l_i')}{|L'|}$$

Note that $m$ is a user defined sample size, used $m = 3000$ for each ontology with a very large number of concepts.

```
flag OntologyType;
int   iSampleSize;
float[iSampleSize] Freq1;   //Freq1 stores the occurrence count.
String[iSampleSize] Candidate; //Candidate will contain concepts.
String sTableName=" SAS_concepts ";

switch (OntologyType)
{ case WordNet:
     iSampleSize = 3000; //Initialize iSampleSize for WordNet
    for (j=1; j<=iSampleSize; j++)
    {   Candidate[j] = getRandomIndexWord(POS.NOUN);
                    // randomly get a word from Index file, noun.index
    }
   break;

   case OpenCyc:
     iSampleSize = 3000; //Initialize iSampleSize for OpenCyc
     for (j=1; j< iSampleSize; j++)
     {   Candidate[j] = getRandomConstant();
                    //randomly get a atomic term from database
     }
   break;

case UMLS_Metathesaurus:
      int  iCUI1, iCUI2;
      iSampleSize = Count the number of concept  pairs  whose relation is 'is-a' in
                    the UMLS table, mrrel
     j=1; k =0;
     while (j<= iSampleSize) {
          (iCUI1, iCUI2) = Read_From_UMLS_Table("mrrel");
                         //Get two concept IDs.
          Candidate[k] = Read_From_UMLS_Table("mrconso", iCUI1); k++;
          Candidate[k] = Read_From_UMLS_Table("mrconso", iCUI2); k++;
                         //Get concept name
          j++;
        }
   break;

case UMLS_SemanticNetwork:
       int  iCUI1, iCUI2;
```

```
    iSampleSize = Count the number of concept pairs whose relation is 'is-a'
                  in the UMLS table, srstr
    j=1; k=0;
    while (j<= iSampleSize)  {
        (iCUI1, iCUI2) = Read_From_UMLS_Table("srstr");
                  //Get two concept IDs.
        Candidate[k] = Read_From_UMLS_Table("mrconso", iCUI1); k++;
        Candidate[k] = Read_From_UMLS_Table("mrconso", iCUI2); k++;
                  //Get concept name
        j++;
    }
  break;

} //end of switch
for (j=1; j< iSampleSize; j++)  {  Freq1[j] = Google#(Candidate[jj]);
                                   //Freq1[j] will contain Google#(X)
                                }
Save_To_Database(Freq1,sTableName); // Save Frequency Array to file
// sTableName is SAS_concepts which is dataset for statistical analysis.
```

**Explanation of Algorithm 5.1**

(1) First, concepts are obtained.

(1a) **WordNet**: The algorithm randomly selects some words from the Index file, noun.index to generate each $X_i$ by using the WordNet API method, *Dictionary.getInstance().getRandomIndexWord (POS.NOUN)* and stores them in an array. Note that, if a noun has many senses, the first sense is selected.

(1b) **OpenCyc**: The algorithm randomly selects atomic terms from the OpenCyc database installed in a local server to generate each $X_i$ by using the *getRandomConstant()* method of the *CycAccess* class in the OpenCyc Java API, and stores them in an array.

(1c) **UMLS Metathesaurus**: The method, *Read_From_UMLS_Table("mrrel")* returns the two *Concept Unique Identifiers* (CUIs) from the "mrrel" table by

accessing the UMLS database. Each obtained CUI will be used to extract its corresponding word label to form an $X_i$ from the "mrconso" table. Note that the function, *Read_From_UMLS_Table(TableName)* uses XML to directly get records from the UMLS database.

(1d) **UMLS Semantic Network**: The method, *Read_From_UMLS_Table("srstr")* returns two *CUIs* from the "srstr" table. In the same way as for the Metathesaurus, word labels of *CUIs* are found.

(2) Function Google#( $X_i$ ) reads the array or the text file and each time queries the Google Server to obtain the frequency of occurrence. Note that, if $X_i$ is 'camelCase', the function will separate it into several independent words and insert a space between any two words.

(3) The algorithm writes all of the search results to a database table, $SAS_{concepts}$ for statistical analysis.

Next, the function *Google# (* $X_i$ ) reads the array or the text file of concepts, and each time queries the Google Server to obtain the frequency of occurrence. Note that, if $X_i$ is 'camelCase', the function will separate it into several independent words and insert a space between any two words. Finally, the algorithm writes all of the search results to a database table for statistical analysis and returns the estimate of concept_naturalness for the ontology in eqn (5.1).

**5.3.2.2 Approximating the Naturalness of IS-A Relationships.** Extracting IS-A relationships depends on APIs and the structure of the ontology. Since not all ontologies provide an explicit list of IS-A relationships, we utilize *m* number of sample concepts

which are produced by Algorithm 1 and find their IS-A relationships.

**Algorithm 5.2: Approximating the naturalness of concept pairs in IS-A relationships**(*String[iSampleSize] Candidate)*

**Input:** A certain ontology $O$, represented as a set of sample concepts $L' = \{l_1', l_2', \cdots, l_m'\}$ with an either explicit set $A$ of IS-A relationships, or a getParent function.

**Output:**

1. $A' = \{(P,C) \mid C \in L' \wedge ((P \Rightarrow C) \in A \vee P = getParent(C))\}$
2. CPGN($A'$), corresponding CPGNs for each IS-A relationship in $A'$.
3. $$\text{naiveISA\_Naturalness}(O) = \frac{\sum_{i=1}^{|A'|} Google\#(P_i \wedge C_i)}{|A'|}$$

Note that $|A'| \le m$.

```
//input: String[iSampleSize] Candidate  where data was  filled by Algorithm1
flag OntologyType;
int   iSampleSize;
float[iSampleSize] FreqPair;
                //FreqPair stores the occurrence count for many concept pairs
class Pairwise_Word{ String Candidate1, Candidate2; }
                // Pairwise_Word.Candidate1 will contain a child concept
                // Pairwise_Word.Candidate2 will contain a parent concept
Pairwise_Word[iSampleSize] CandidatePair;
                //CandidatePair will contain many concept pairs each connected
                //by an IS-A relationship
String sTableName=" SAS_IS-AConceptPair ";
switch (OntologyType)
{ case WordNet:
    for (j=1; j< iSampleSize; j++)
    { CandidatePair[j].Candidate1 = Candidate[j];
            //Candidate[j] contained a child concept in Algoritm1
     CandidatePair[j].Candidate2 =
                                getParent(CandidatePair[j].Candidate1);
     // getParent(argument) calls
     //PointerUtils.getInstance().getDirectHypernyms(argument) API method
     }
   break;

   case OpenCyc:
    for (j=1; j< iSampleSize; j++)
     {  CandidatePair[j].Candidate1 = Candidate[j];;
        CandidatePair[j].Candidate2 =
```

```
                                    getParent(CandidatePair[j].Candidate1);
        // getParent(argument) calls getGenls(argument) OpenCyc API method
    }
  break;

  case UMLS_Metathesaurus:
     int iCUI1, iCUI2;
     for (j=1; j< iSampleSize; j++)  {
      (iCUI1,iCUI2) = getConceptPairIDs_from_UMLS_Table ("mrrel","is-a");
      CandidatePair[j] =
                    getConceptPair_from_UMLS_Table
  ("mrconso",iCUT1,iCUT2);
                     //Get concept names
     }
  break;

  case UMLS_SemanticNetwork:
     int iCUI1, iCUI2;
     for (j=1; j< iSampleSize; j++)  {
      (iCUI1,iCUI2) = getConceptPairIDs_From_UMLS_Table("srstr","is-a");
      CandidatePair[j] =
                    getConceptPair_from_UMLS_Table
  ("mrconso",iCUT1,iCUT2);
                     //Get concept names
     }
  break;
} //end of switch
for (j=1; j< iSampleSize; j++)   {   FreqPair [j] = CPGN(CandidatePair[j]);
                                      // FreqPair[j] contains Google#(X_i ∧ Y_j)
                                    }
 Save_To_Database(FreqPair, sTableName);
 // sTableName is SAS_{IS-AConceptPair} which is dataset for statistical analysis
```

**Explanation of Algorithm 5.2**

Concept pairs are obtained, taking an input array which contains the randomly sampled concepts in the Algorithm 1:

(1) Concept pairs are obtained by taking an input array which contains the randomly sampled concepts in the Algorithm 1.

(1a) **WordNet**: The algorithm reads the array to define each $X_i$, and obtains its

parent, $Y_j$ for each $X_i$ by using the function *getParent* $(X_i)$. It calls the *PointerUtils.getInstance().getDirectHypernyms()* WordNet API method. The algorithm stores all concept pairs $(X_i, Y_j)$ in an array.

$A' = \{(P,C) \mid C \in L' \wedge P = getParent(C)\}$

(1b) **OpenCyc**: In a similar way as for WordNet, the algorithm reads the array to define each $X_i$ and obtains its $Y_j$ for each $X_i$ by using $getParent(X_i)$. It calls the *getGenls()* OpenCyc API method. The algorithm stores all concept pairs $(X_i, Y_j)$ in an array. $A' = \{(P,C) \mid C \in L' \wedge P = getParent(C)\}$

(1c) **UMLS Metathesaurus**: The function, *getConceptPairIDs_from_UMLS_Table ("mrrel","is-a")* returns two CUIs which are in IS-A relationships from the UMLS table, "mrrel". A pair of word labels of the two *CUIs* $(X_i, Y_j)$ is obtained in the same way as in the Algorithm 5.1.

$A' = \{(P,C) \mid C \in L' \wedge (P \Rightarrow C) \in mrrel\}$

(1d) **UMLS Semantic Network**: The method, *getConceptPairIDs_from_UMLS_Table ("srstr", "is-a")* returns two *CUIs* which are in IS-A relationships from the UMLS table, "srstr". A pair of word labels of the two *CUIs* $(X_i, Y_j)$ is obtained in the same way as in the Algorithm 5.1.

$A' = \{(P,C) \mid C \in L' \wedge (P \Rightarrow C) \in srstr\}$

(2) Function *CPGN* returns the frequencies after sending each concept pair $(X_i, Y_j)$ to the Google server.

(3) The algorithm writes the search results to a database table, $SAS_{IS-AConceptPair}$ for statistical analysis and returns the estimate of ISA_naturalness for the ontology in Equation (5.2) or (5.9).

**5.3.2.3 Approximating the Naturalness of Semantic Relationships.** Semantic relationships are provided by ontologies in different ways; while some ontologies provide explicit table, $T$ listing all binary relationships, others provide a list of API functions, $F$. When an explicit tables, $T$ of all binary relationships is given, $CP'$, a sample set of concept pairs with semantic relationships, can be easily generated by a random selection excluding the IS-A or inverse IS-A relationships. Note that $T$'s schema includes a concept $L$, a concept $R$ and a *relationship name*, etc. where $L$ and $R$ are connected by the relationship. The list of semantic relationships includes *IS-A, inverse IS-A,* etc. Examples of this kind of ontology are the UMLS Metathesaurus and the UMLS Semantic Network. Alternatively, the API function set, $F = (f_1, f_2, \cdots, f_n)$ may be given such that each $f_i$ represents are direction of a relationship. We use a function, $CP_i = \text{get\_concept\_pairs}(f_i)$ which takes $f_i$ as an input and outputs $CP_i$, a set of all concept pairs consisting of *Ls* and *Rs* of this relationship. This kind of ontology is exemplified by OpenCyc. Lastly, when $F$ is given without a get_concept_pairs feature, one must try different *R*s to extract its corresponding $L$ exhaustively such that $L = f_i(R)$. An example of this kind of ontologies is WordNet .

**Algorithm 3: Approximating the naturalness of concept pairs in semantic relationships**

**Input:** A relationship table $T = (L, R, relationship)$ of a certain ontology $O$ or an API function set, $F$ of a certain ontology $O$.

**Output:**

1. $CP' = sampling(CP)$ where

   $CP = \{(L,R) \mid (L,R,relationship) \in T \wedge relationship \neq$ 'IS_A' or 'reverse IS_A'

   or $CP = \{(L,R) \mid f_i \in F \wedge L = f_i(R) \wedge f_i \neq$ 'IS_A' or 'reverse IS_A'$\}$

   or $CP = \{(L,R) \mid f_i \in F \wedge f_i(relationship)$ is true $\wedge (L,R) = f_i(reletionship)\}$

2. CPGN($CP'$), that is, the corresponding CPGNs for each semantic relationship in $CP'$.

3. approximate semantic_Naturalness$(O) = \dfrac{\sum_{i=1}^{|CP'|} Google\#(L_i, R_i)}{|CP'|}$

```
flag OntologyType;
int   iSampleSize;
float[iSampleSize] FreqPair;
     // FreqPair stores the occurrence count for many concept pairs
class Pairwise_Word{ String Candidate1, Candidate2; }
Pairwise_Word[iSampleSize] CandidatePair;
    //CandidatePair will contain many concept pairs each connected by
    //a Semantic  relationship
Pairwise_Word[ ] BinaryPreList;
   // BinaryPreList contains concept pairs connected by OpenCyc binary
   //predicates
Enumeration WordNetSemanticRelationType {Causes, EntailedBy,  Entailments,
      PartHolonyms, PartMeronyms, SubstanceHolonyms,  SubstanceMeronyms
};
WordNetSemanticRelationType WNSR;
String sTableName= " SAS SemanicConceptPair ";
switch (OntologyType)
{ case WordNet:
     String sTempCandidate;  // sTempCandidate  contains a random word.
                 //The word will be saved when it has any  semantic relationship.
    j=1; k=0;
    iSampleSize= 200;
    while (j <= iSampleSize)
```

```
{ sTempCandidate = randomly get a word from Index file(noun.index);
        for each semantic relationship listed in WNSR{
                relConcept =
                        getConceptInSemanticRelation(sTempCandidate);
                if (relConcept!=NULL) {
                        CandidatePair[k].Candidate1 = sTempCandidate;
                        CandidatePair[k].Candidate2 = relConcept;
                        //sTempCandidate is related with relConcept  by a
                    semantic //relationship
                        k++; }
        } j++;
  }
 break;
case OpenCyc:
  String sTempCandidatePre;
            // sTempCandidatePre contains a random atomic term
  j=1; k =0;
  iSampleSize= 200;
  while (j <= iSampleSize)
  { sTempCandidatePre = randomly get an atomic term from database;
       if sTempCandidatePre is 'binary predicate'{
            l=0;
            BinaryPreList =
                    getConceptPairSemanticRelation(sTempCandidatePre);
                  // binary predicate relates one concept to another concept
            for each pair in BinaryPreList{
                CandidatePair[j]= BinaryPreList[l];  l++;
            k++; }
      } j++;
   }  //end of while
 break;

case UMLS_Metathesaurus:
  int  iCUI1, iCUI2;
  iSampleSiz =Count the number of concept  pairs  whose relation is 'non-is-a'
          in the UMLS table mrrel
  for (j=1; j< iSampleSize; j++)  {
    (iCUI1,iCUI2) =
              getConceptPairIDs_from_UMLS_Table ("mrrel","non-is-a");
    CandidatePair[j] =
            getConceptPair_from_UMLS_Table ("mrconso",iCUT1,iCUT2);
                //Get concept names
  }
 break;

case UMLS_SemanticNetwork:
```

```
    int iCUI1, iCUI2;
    iSampleSize=Count the number of concept pairs whose relation is 'non-is-a'
                in the UMLS table srstr
    for (j=1; j< iSampleSize; j++)  {
      (iCUI1,iCUI2) =
              getConceptPairIDs_From_UMLS_Table("srstr","non-is-a");
      CandidatePair[j] =
            getConceptPair_from_UMLS_Table ("mrconso",iCUT1,iCUT2);
    }
  break;
}//end of switch
for (j=1; j< iSampleSize; j++)   { FreqPair [j] = CPGN(CandidatePair[j]); }

Save_To_Database(FreqPair, sTableName);
// sTableName is SAS_SemanicConceptPair which is dataset for statistical analysis.
```

**Explanation of Algorithm 5.3**

(1) Concept pairs connected by semantic relationships are obtained.

(1a)**WordNet**: For each semantic relationship defined in WordNetSemanticRelationType, the algorithm checks whether a randomly selected concept $X_i$ has one of the defined semantic relationships using *getConceptInSemanticRelation(concept)*. It calls *getPartHolonyms()*, *getAttributes()*, *getPartMeronyms()*, and *getSubstanceHolonyms()*, *getCauses()*, *getEntailedBy()*, *getEntailments()* in the API to obtain a concept pair $(X_i, Y_j)$. The algorithm stores all concept pairs in an array.

(1b) **OpenCyc**: The algorithm randomly gets an atomic term from the OpenCyc database. If the term is a binary predicate, it calls the *getConceptPairSemanticRelation(term)* function. The function calls the *getInterArgIsa1_2s(term)* method of the OpenCyc API and returns the list of concept pairs connected by the binary predicate. For each concept pair $(X_i, Y_j)$,

the algorithm stores all concept pairs in an array.

(1c) **UMLS Metathesaurus**: Similar functions as for IS-A relationships are used to extract concept pairs $(X_i, Y_j)$ for semantic relationships. However, a parameter "non-is-a" is used. The complete list of the semantic relationships for the Metathesaurus is in the Appendix.

(1d) **UMLS Semantic Network**: Similar functions as for IS-A relationships are used to extract concept pairs $(X_i, Y_j)$ for semantic relationships. However, a parameter, "non-is-a" is used. The complete list of the semantic relationships for the Semantic Network is also in the Appendix.

(2) Function *CPGN* returns the Google frequency.

(3) The algorithm writes the search results, $Google\#(X_i \wedge Y_j)$ to a database table, $SAS_{SemanicConceptPair}$ for statistical analysis.

### 5.3.3 Phase II: Analyze Data

For the statistical comparison analysis, a t-test and an ANOVA test were used. Statistical parameters of two populations are highly unlikely to be identical, and statistical methods allow us to decide whether two apparently different values are indeed significantly different. In this dissertation work, the t-statistic (pooled and Sattherwaite) is used to determine whether significant differences exist between pairs or groups of ontologies.

**t-test:** The t-test is adopted to compare the means of two groups. Let $\bar{x}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} x_{1i}$

$\bar{x}_2 = \frac{1}{n_2}\sum_{i=1}^{n_2} x_{2i}$ be the means of the sampling units of two groups where, $n_1$ and $n_2$ are

the numbers of elements of the sampling units in each of the two groups.

From the above means, the variances of the sampling units in each of the two groups are computed as follows:

$$s_1^2 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}(x_{1i} - \bar{x}_1)^2 \quad s_2^2 = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}(x_{2i} - \bar{x}_2)^2 \tag{5.10}$$

If the population variances $\sigma_1^2$ and $\sigma_2^2$ of the two groups are not known but the two populations are assumed to be the same, that is, $\sigma_1^2 = \sigma_2^2 (= \sigma^2)$, the distribution of the difference of the means in the sampling units of two groups conforms to the normal distribution with its mean and variance, denoted by $E(\bar{x}_1 - \bar{x}_2) = \mu_1 - \mu_2$ and $Var(\bar{x}_1 - \bar{x}_2) = \sigma^2(1/n_1 + 1/n_2)$ respectively. Note that the difference between the means in the two populations, $\mu_1 - \mu_2$, to which the two sampling units belong, is estimated by the difference in the means in the two sampling units. Therefore, the pooled estimator of the common population variance $\sigma^2$ is denoted by $S_p^2$ and computed as follows (University of California 2006):

$$S_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \tag{5.11}$$

From the formulas presented earlier especially $S_p^2$, the t-statistic is computed as follows:

$$t - statistic = \frac{(\bar{x}_1 - \bar{x}_2) - \delta_0}{S_p\sqrt{1/n_1 + 1/n_2}} \tag{5.12}$$

where $\delta_0$ is the difference between the means in the two populations. If $\delta_0 = 0$, the null hypothesis $H_0 : \mu_1 - \mu_2 = 0$ that is, there is no difference between the means, is examined.

On the other hand, when the two populations are not assumed to be the same, that is, $\sigma_1^2 \neq \sigma_2^2$, the distribution of the means of the sampling units of two groups does not conform to the normal distribution. In this case, if the degree of freedom (*df*) is changed from $df = n_1 + n_2 - 2$ to $df = \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{(s_1^2/n_1)^2/(n_1-1) + (s_2^2/n_2)^2/(n_2-1)}$, the distribution of the difference in the means in the sampling units of two groups asymptotically conforms to the t-distribution. The Satterthwaite method adopts this modification.

**ANOVA (Analysis of Variances) test**: In order to compare two or more means, the ANOVA test is employed. The variance of the data (i.e. *Google#*) is divided into two sources from which the variance originates (University of California 2006): in this research model (a) the variance between groups (i.e. ontologies) or conditions (i.e. different label lengths) (b) the variance within groups and conditions.

Let $y_{ij}$ be the value which the j-th observation among $n_i$ observations shows in response to the i-th group or condition. The condition or group mean of the i-th condition to which $y_{ij}$ belongs is computed as $\bar{y}_{i.} = \sum_{j=1}^{n_i} y_{ij}/n_1$, and "the sum of squares of deviations about the mean" are computed as $\sum_{j=1}^{n_i}(y_{ij} - \bar{y}_{i.})^2$.

The function of ANOVA is to test the null hypothesis namely that the population mean is the same in all conditions or groups. To test the hypothesis, the f-statistic is computed by the following equation.

$$\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(y_{ij}-\bar{y}_{..}\right)^2=\sum_{i=1}^{k}n_i\left(\bar{y}_{i.}-\bar{y}_{..}\right)^2+\sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\bar{y}_{i.})^2 \tag{5.13}$$

where $k$ is the number of groups or conditions and $\bar{y}_{..}$ is the mean of the condition means. The above equation (5.13) can be rewritten by using the standard abbreviation (University of California 2006) :

Total sums of squares (TSS) = between-conditions sum of squares (=BSS) + within-conditions sum of squares (=WSS) (5.14)

TSS indicates the total variation such that when BSS is very high compared with WSS, it can be inferred that TSS comes from the variance between the mean values among different conditions or groups. That is, there is the influence of conditions or groups on the values of observations.

$$f-statistic=\frac{BSS/(k-1)}{WSS/(n-k)} \quad \text{where } n=\sum_{i=1}^{k}n_i \tag{5.15}$$

The f-statistic is a ratio of the $BSS/(k-1)$ divided by the $WSS/(n-1)$. A large f-statistic is evidence that the null hypothesis may be rejected, since it tells that the variance of the data mostly originates from the difference between conditions/groups rather than within conditions/groups. The ANOVA determines significance probability from the f-statistic.

In computing the t-statistic and f-statistic described, the *SAS* (statistical analysis

software) is used and the results are interpreted based on the related theories. The detailed results are presented in the following Section 5.4.

## 5.4 Experimental Results on Existing Large Ontologies

### 5.4.1 The Naturalness of Concepts

**Descriptive Statistics**

Following are the symbols used in this dissertation and their corresponding statistical measurements: "M", the mean value of the number of search results for a concept, "SD", the standard deviation, "R", the Range (the difference between the minimum and the maximum), "K", Kolmogorov-Smirnov (a statistical method for testing normality). "N" is the sample size.

Following are the symbols used in this paper for ontologies: "W" = WordNet,"US" = UMLS Semantic Network, "UM" = UMLS Metathesaurus, "OCC" = OpenCyc Class, "OCI" = OpenCyc Individual, and "OC" = OpenCyc (complete). In Section 5.4.2, the abbreviation "UMS"=a set of concept pairs (X, Y) shall be also used where X is in UMS and Y is in US.

**Table 5.1** The Descriptive Statistics I: Concept Occurrence

| | W | US | UM | OC | OCC | OCI |
|---|---|---|---|---|---|---|
| **N** | 3,000 | 135 | 4,858 | 3,000 | 1,434 | 1,566 |
| **M** | 837,584 | 5,059,901 | 21,499 | 540,426 | 771,400 | 328,921 |
| **SD** | 6,841,168 | 16,622,855 | 313,416 | 5,186,753 | 6,983,312 | 2,608,801 |
| **R** | 195,999,948 | 116,999,967 | 16,800,000 | 182E6 | 182E6 | 6.805584E12 |

Table 5.1 shows the descriptive statistics for concepts. The ontology with the highest mean value, 5,059,901 is the Semantic Network and the ontology with the lowest mean value, 21,499 is the Metathesaurus of the UMLS. This agrees with the intuition, as the Metathesaurus contains many highly specialized medical terms. The result of the

*Kolmogorov-Smirnov test*, which is a common normality distribution shows that the search results from Google are not normally distributed and consequently, the results of the unequal variances t-statistic are emphasized since the test for equal variances is highly sensitive to non-normality.

**Comparision of Means in Two Independent Ontologies**

Following are the symbols used in this paper and their corresponding statistical measurements: "DF", the degrees of freedom, "t Value", the t-statistic, "Pr>|t|", the probability of a t-value in which the mean value of equal or greater absolute value conforms to the null hypothesis (University of California 2006).

**Table 5.2** The T-test for the Difference of Means between *OpenCyc* vs. *WordNet* for Concepts

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 5998 | -1.90 | 0.0580 |
| **Satterthwaite /Unequal** | 5590 | -1.90 | 0.0580 |

Table 5.2 shows the difference of means between WordNet vs. OpenCyc. The value of unequal t is –1.90, and the p-value is 0.0580. The null hypothesis of equal mean values can be rejected at the 10% level. It can be concluded that there is significant difference between the means of OpenCyc and WordNet. Looking at descriptive statistics in Table 5.1, it can be further concluded that the average number of search results for WordNet is significantly higher than for OpenCyc, meaning that WordNet concepts are more natural than OpenCyc concepts.

Table 5.3 shows the difference of means between OpenCyc Class vs. WordNet. The value of unequal t is –0.30, and the p-value is 0.7644. At the 5% level, the two means are not significantly different. The null hypothesis of equal means cannot be rejected at

the 5% level. It can be concluded that there is no significant difference between the means of WordNet and OpenCyc Class.

**Table 5.3** The T-test for the Difference of Means between *OpenCyc Class* vs. *WordNet* for Concepts

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 4432 | -0.30 | 0.7647 |
| **Satterthwaite /Unequal** | 2771 | -0.30 | 0.7664 |

Table 5.4 shows the difference of means between OpenCyc Individual vs. WordNet. The value of unequal t is –3.60, and the p-value is 0.0003. At the 5% level, the two means are significantly different. The null hypothesis of equal means can be rejected at the 5% level. It can be concluded that the average number of search results for WordNet is significantly higher than for OpenCyc Individual.

**Table 5.4** The T-test for the Difference of Means between *OpenCyc Individual* vs. *WordNet* for Concepts

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 4564 | -2.84 | 0.0046 |
| **Satterthwaite /Unequal** | 4268 | -3.60 | 0.0003 |

Table 5.5 shows the difference of means between the UMLS Semantic Network vs. WordNet. The value of unequal t is 2.94, and the p-value is 0.0039. At the 5% level, the two means are significantly different. The null hypothesis of equal means can be rejected at the 5% level. It can be concluded that the average number of search results for the UMLS Semantic Network is significantly higher than for WordNet.

**Table 5.5** The T-test for the Difference of Means between *UMLS Semantic Network* vs. *WordNet* for Concepts

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 3133 | 6.38 | 0.0001 |
| **Satterthwaite /Unequal** | 136 | 2.94 | 0.0039 |

Table 5.6 shows the difference of means between UMLS Metathesaurus vs. WordNet. The value of unequal t is -6.53, and the p-value is 0.0001. At the 5% level, the two means are significantly different. The null hypothesis of equal means can be rejected at the 5% level. It can be concluded that the average number of search results for WordNet is significantly higher than for UMLS Metathesaurus.

**Table 5.6** The T-test for the Difference of Means between *UMLS Metathesaurus* vs. *WordNet* for Concepts

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 7856 | -8.30 | 0.0001 |
| **Satterthwaite /Unequal** | 3007 | -6.53 | 0.0001 |

**Analysis of the Means for three Groups of Ontologies**

Table 5.7 shows that the search results from Google are significantly different in the investigated ontologies. There are 10,858 labels as samples from the ontologies. The F-statistic is 32.55. Because the p-value is small, the null hypothesis of equal means for the different ontologies is rejected. The conclusion is that the numbers of search results returned by Google for concepts from different ontologies are different.

The result of *Tukey comparisons* (TK) which were used to further investigate the differences in the search results is shown in Table 5.7. Three groups, **A**, **B** and **C** are shown below the means. The mean of search results for the different ontologies are significantly different. WordNet is in group **A**. OpenCyc is in group **B**. The Metathesaurus of the UMLS is in **C**. That is, the search results between these three groups, **A**, **B** and **C** are significantly different. Note that the labels from the Semantic Network of the UMLS were removed in this test since its size is very small compared to other ontologies.

**Table 5.7** Results of the ANOVA Test to Examine the Variance among the Different Ontologies for Concepts

| | W | OC | UM | F | Pr > F |
|---|---|---|---|---|---|
| N | 3,000 | 3,000 | 4,858 | 32.55 | 0.0001 |
| M | 837,584 | 540,426 | 21,499 | | |
| TK | A | B | C | | |

**Analysis of the Influence of Label Length on Naturalness**

Table 5.8 shows that the number of the search results is significantly different according to the number of words in a label. That implies there is one factor, label length, with five levels, having values 1, 2, 3, 4 and 5, which needs to be taken into consideration. As a reminder, the number of words in a concept label is measured as label length. There are 10,406 observations. The F-statistic for testing whether label length is significant is 56.78. Because the p-value is small, the null hypothesis of equal means for the different label lengths is rejected. The conclusion is that the results from Google are different with respect to the different lengths of the labels.

Table 5.8 also shows the results from the Tukey comparisons. Two groups, A and B are shown below the means. The conclusion is that the means for the different label lengths are significantly different. The labels of length 1 are significantly different from labels with the lengths, 2, 3, 4 and 5 respectively. On the other hand, from the length 2 and up, labels are not significantly different.

**Table 5.8** Results of the ANOVA Test to Examine the Variance between Multi-Word labels and Single-Word labels for Concepts

| | single- word | 2-word | 3-word | 4-word | 5-word | F | Pr > F |
|---|---|---|---|---|---|---|---|
| M | 1,731,209 | 85,815 | 9,283 | 3,330 | 448 | 56.7 | <0.000 |
| N | 2,640 | 3,817 | 2,071 | 1,171 | 707 | 8 | 1 |
| TK | A | B | B | B | B | | |

### 5.4.2 The Naturalness of Concept Pairs in IS-A Relationships

**Descriptive Statistics**

An additional set of pairs (X, Y) can be defined where X is derived from the concepts of the Metathesaurus and Y from the Semantic Types of the Semantic Network. The notation UMS for these pairs is used. All other abbreviations are the same as before.

Table 5.9 shows the descriptive statistics. The variable is the number of search results when a user queries a concept pair to Google. The ontology with the highest mean value, 323,777 is the Semantic Network. On the other hand, the relationship between the Semantic Network and the Metathesaurus of the UMLS has the lowest mean value, 62,138.

**Table 5.9** Descriptive Statistics II: Pair Occurrence in IS-A Relationships

| | W | US | UM | UMS | OCC |
|---|---|---|---|---|---|
| **N** | 3091 | 135 | 5,000 | 6,249 | 7,787 |
| **M** | 86,643 | 323,777 | 3,752 | 1,410 | 7,867 |
| **SD** | 1,094,492 | 1,774,338 | 57,133 | 62,138 | 184,993 |
| **R** | 55,099,969 | 17,699,975 | 2,500,000 | 4,500,000 | 342224E10 |

**Comparison of Means in Two Independent Ontologies**

In this section, the natural association of concept pairs is analyzed. This section typically concentrates on the results which are interestingly different from the results of the naturalness of a concept.

Table 5.10 shows the difference of means between OpenCyc Class vs. WordNet. The value of unequal t is –4.63, and the p-value is 0.0001. At the 5% level, the two means are significantly different. The null hypothesis of equal means can be rejected at the 5% level. It can be concluded that the average number of the degree of natural associations for WordNet is significantly higher than for OpenCyc.

**Table 5.10.** The T-test for the Difference of Means between *OpenCyc Class* vs. *WordNet* for IS-A pairs

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 11E3 | -7.23 | 0.0001 |
| **Satterthwaite /Unequal** | 3132 | -4.63 | 0.0001 |

Table 5.11 shows the difference of means between UMLS Semantic Network vs. WordNet. The value of unequal t is 1.03, and the p-value is 0.3037. At the 5% level, the two means are not significantly different. The null hypothesis of equal means cannot be rejected at the 5% level. It can be concluded that there is no significant difference between the means of the UMLS Semantic Network and WordNet.

**Table 5.11** The T-test for the Difference of Means *UMLS Semantic Network* vs. *WordNet* for IS-A pairs

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 3224 | 1.27 | 0.2024 |
| **Satterthwaite /Unequal** | 141 | 1.03 | 0.3037 |

Table 5.12 shows the difference of means between OpenCyc vs. the UMLS Metathesaurus. The value of unequal t is –0.10 and the p-value is 0.9211. At the 5% level, the two means are not significantly different. The null hypothesis of equal means can not be rejected at the 5% level. It can be concluded that there is no significant difference between the means of OpenCyc Class and UMLS Metathesaurus.

**Table 5.12** The T-test for the Difference of Means between *OpenCyc Class* vs. *UMLS Metathesaurus* for IS-A pairs

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 13,000 | -0.10 | 0.9217 |
| **Satterthwaite /Unequal** | 11,000 | -0.10 | 0.9211 |

Table 5.13 shows the difference of means between UMS vs. WordNet. The value of unequal t is –4.92 and the p-value is 0.0001. At the 5% level, the two means are significantly different. The null hypothesis of equal means can be rejected at the 5%

level. It can be concluded that the average number of the degree of natural associations for WordNet is significantly higher than for the relationships that hold between the Metathesaurus and Semantic Network (which was called UMS before).

**Table 5.13** The T-test for the Difference of Means between *UMS* vs. *WordNet* for IS-A pairs

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 9,338 | -6.97 | 0.0001 |
| **Satterthwaite /Unequal** | 3,106 | -4.92 | 0.0001 |

Table 5.14 shows the difference of means between OpenCyc Class vs. UMS. The value of unequal t is 2.89 and the p-value is 0.0038. At the 5% level, the two means are significantly different. The null hypothesis of equal means can be rejected at the 5% level. It can be concluded that the average number of the degree of natural associations for OpenCyc Class is significantly higher than for UMS.

**Table 5.14** The T-test for the Difference of Means between *OpenCyc Class* vs. *UMS* for IS-A pairs

| Method/Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|
| **Pooled/ Equal** | 14,000 | 2.73 | 0.0064 |
| **Satterthwaite /Unequal** | 13,000 | 2.89 | 0.0038 |

### 5.4.3 The Naturalness of Concept Pairs by Semantic Relationships

Table 5.15 shows the descriptive statistics for concepts. The ontology with the highest mean value, 1,079,738 is the Semantic Network and the ontology with the lowest mean value, 2,759 is the Metathesaurus of the UMLS.

**Table 5.15** The Descriptive Statistics III: Pair Occurrence in Some Semantic Relationships

| | W | US | UM | OC |
|---|---|---|---|---|
| **N** | 1,095 | 316 | 2,171 | 414 |
| **M** | 949,976 | 1,079,738 | 2,759 | 157,900 |
| **SD** | 5,965,384 | 8,323,491 | 29,666 | 1,443,433 |
| **R** | 158,999,952 | 113,999,974 | 667,000 | 27,400,000 |

The results of Tukey comparisons (TK) which were used to investigate the differences in the naturalness of semantic relationships among the ontologies is shown in Table 16. Two groups, A and B are shown below the means. The mean values of Google numbers for concept pairs in semantic relationships among the different ontologies are significantly different. The Semantic Network of the UMLS and the WordNet are in group A while the OpenCyc and the Metathesaurus of the UMLS are in group B.

**Table 5.16** Results of the ANOVA Test to Examine the Variance among the Different Ontologies for Semantic Relationships

| | US | W | OC | UM | F | Pr > F |
|---|---|---|---|---|---|---|
| **N** | 316 | 1,095 | 414 | 2,171 | 18.11 | 0.0001 |
| **M** | 1,079,738 | 949,976 | 157,900 | 2,759 | | |
| **TK** | **A** | **A** | **B** | **B** | | |

### 5.4.4 Overall Ranking

**Naturalness by concepts**: In summary, (1) the UMLS Semantic Network is the most natural followed by (2) WordNet and OpenCyc class (3) OpenCyc(Complete) (4) OpenCyc Individual, and (5) the UMLS Metathesaurus. These results are supported by t-test and ANOVA test.

**Naturalness by IS-A pairs (approach 1)**: The ontologies with the largest naturalness are (1) the UMLS Semantic Network and (2) WordNet, followed by (3) OpenCyc Class, and (4) the UMLS Metathesaurus. (5) The associations between Semantic Network Semantic Types and Metathesaurus concepts (UMS) were found to be the least natural concept pairs. Numbering is based on mean of frequency. Thus, the rank is less well established than the rank in naturalness by concepts.

**Naturalness by IS-A pairs (approach 2)**: Concept pairs in IS-A relationships, were evaluated based on rule mining techniques. The likely co-occurrence of a parent

with a child was interpreted as evidence that this child parent pair appears natural to humans.

Several t-tests were conducted to test whether the difference of means of two groups of ontologies is statistically significant with respect to the degree of natural association of concept pairs. In the results, (a) WordNet has a significantly higher naturalness than OpenCyc Class. (b) However, there is no significant difference between the naturalness of WordNet and the Semantic Network of the UMLS. (c) There is no significant difference between the naturalness of OpenCyc Class and the Metathesaurus of the UMLS. (d) WordNet has a significantly higher naturalness than the associations between Metathesaurus and Semantic Network. (e) OpenCyc Class has a significantly higher naturalness than the associations between Metathesaurus and Semantic Network. The results of approach 2 show that some distinctions shown by approach 1 are presumably not significant.

**Naturalness by pairs with semantic relationships:** The (1) Semantic Network shows the highest naturalness followed by (2) WordNet, (3) OpenCyc and (4) the Metathesaurus. Numbering is based on mean of frequency. Thus, the rank is less well established than the rank in naturalness by concepts.

Figure 5.4 shows how natural each ontology is in three segments; concepts, concept pairs connected by IS-A relationships and concept pairs connected by semantic relationships. Each value in the y-axis, which measures naturalness, is based on the descriptive statistics presented in Tables 5.1, 5.9 and 5.15.

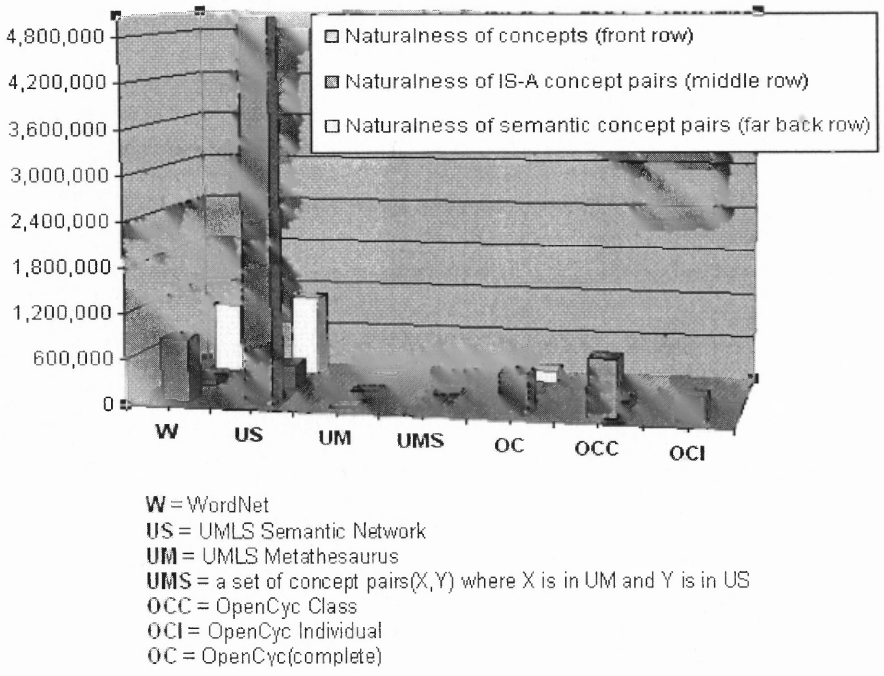



**Figure 5.4** Descriptive graph to show the naturalness of ontologies.

Comparing the results of all three naturalness measures (concepts, IS-A pairs, and semantic pairs) they agree substantially. Thus, no complicated scheme was needed to aggregate results.

**Overall Naturalness:** The most natural ontology component is the (1) Semantic Network, followed by (2) WordNet, (3) OpenCyc Class, (4) OpenCyc (Complete) and (5) OpenCyc Individual. The least natural ontology (component) is (6) the Metathesaurus. The Semantic Network is the best, but the number of concepts in it is comparatively very small, too small to be useful in practice. It can be concluded that WordNet, which is the second most natural ontology, can be used to define a likely upper limit or at least a reference point in evaluating naturalness of other large ontologies. This result was expected, since it consists of words in "human lexical memory" by design.

## 5.5 Experimental Results for Naturalness of Domain Ontologies

The domain ontologies in eight domains (i.e., Airfares, Automobiles, Books, Car Rentals, Hotels, Jobs, Movies and MusicRecords) were automatically generated in this dissertation. In this section, an evaluation of these eight domain ontologies from the naturalness perspective is presented.

It should be noted that there were some restrictions because of the query limit policy of the Google company. The evaluation for naturalness of existing ontologies such as WordNet, OpenCyc and the UMLS, i.e., measuring the Google number, $Google\#(c_i)$ was conducted successfully, as special permission for an increased query limit was granted by the Google company. Evaluation of the naturalness of the eight domain ontologies was conducted in June 2007 when the special permission had expired. Hence, the remote server of the Google company produced frequent disconnection errors and required long processing times. Consequently, not all the concepts could be sent to the Google server. Thus, only sampled concepts were processed and will be presented here.

Table 5.17 shows the descriptive statistics for concepts appearing in the domain ontologies. The naturalness of concepts in all domain ontologies is higher than the naturalness of Semantic Network, which has the highest mean value, 5,059,901. Thus, it can be concluded that concepts in the domain ontologies are more natural than these in the existing ontologies that were analyzed in this research.

**Table 5.17** Naturalness of Concepts in Domain Ontologies

| | **Airfares** | **Automobiles** | **Books** | **Car Rentals** |
|---|---|---|---|---|
| ***N* (sample size)** | 89 | 128 | 102 | 108 |
| ***M* (mean)** | 238,656,258 | 353,817,696 | 472,836,581 | 397,810,759 |
| ***SD* (standard deviation)** | 289,318,711 | 398,375,998 | 472,206,657 | 454,913,333 |
| ***R* (range = max – min)** | 1349,982,600 | 2,139,945,200 | 1929,982,700 | 2,009,318,000 |
| | **Hotels** | **Jobs** | **Movies** | **Music Records** |
| **N** | 100 | 104 | 132 | 124 |
| **M** | 253,819,140 | 409,333,236 | 520,390,752 | 272,175,483 |
| **SD** | 275,729,375 | 448,886,776 | 461,098,734 | 365,564,519 |
| **R** | 1,159,872,000 | 2,139,948,500 | 2,069,983,700 | 1,589,954,200 |

Table 5.18 shows the descriptive statistics for naturalness of the IS-A relationships which was defined as eqn 5.2 in Section, 5.2.2. The naturalness of IS-A relationships is higher than in the UMLS Semantic Network which has the highest mean value, 323,777. Thus, it can be also concluded that IS-A relationships in the domain ontologies are more natural than these in the existing ontologies that were analyzed in this research.

**Table 5.18** Naturalness of IS-A Relationships in Domain Ontologies

| | **Airfares** | **Automobiles** | **Books** | **Car Rentals** |
|---|---|---|---|---|
| **N** | 41 | 59 | 62 | 54 |
| **M** | 12,397,485 | 97,826,597 | 188,280,289 | 79,483,481 |
| **SD** | 31,880,518 | 4,100,000 | 34,200,000 | 124,934,017 |
| **R** | 190,999,288 | 577,999,394 | 1,659,982,700 | 621,883,000 |
| | **Hotels** | **Jobs** | **Movies** | **Music Records** |
| **N** | 52 | 66 | 105 | 53 |
| **M** | 51,711,423 | 242,690,879 | 120,002,149 | 32,146,652 |
| **SD** | 119,330,636 | 349,894,552 | 172,064,131 | 89,371,424 |
| **R** | 767,891,000 | 1,909,999,288 | 1,449,981,400 | 500,999,217 |

The naturalness of semantic relationships was not analyzed due to the small amount of data available.

## 5.6 Limitations

One may argue against the reliability of the Internet as corpus by pointing out that there are many noise effects. This is why the research (Brewster *et al.*, 2003) does not consider the Internet as ideal but accepts it as the best available source. Search results can not distinguish terms which have the same spelling but their meaning is different. Sampling 10 pairs of concepts and searching for those on the Internet, it is hard to trust relevant Web sites. Considering the reliability of the Internet sources, the total number of 13,993 concepts, 22,262 concept pairs in IS-A relationships and 3,996 concepts pairs in semantic

relationships were sampled. Statistical mean comparisons for evaluating ontologies were used to reduce the effect of this problem.

Seven components of three important ontologies have been evaluated in this dissertation. One may argue against the selection of the ontologies by pointing out that the comparative approach would be more proper if two ontologies are in the same domain. Several ontologies in the same domain, "university student" were ranked in (Alani and Brewster 2005). However, it is questionable whether ranking of ontologies with small numbers of concepts is useful and reliable. There are few domains that have well developed ontologies of comparable size. Unfortunately, most ontologies available on the Web are not well developed (Ding *et al.* 2004). Thus the best known ontologies from different domains were used. The main criteria in selecting ontologies were whether they are actively used by other applications, whether an API to access their data exists and whether they are of substantial size.

One limitation of the analysis in this study is in the normality assumption for the t-test robustness. As variables did not follow a normal distribution, the results from the Satterthwaite method were alternatively presented which gives t-statistics in an asymptotic way. In addition, the sample size was enlarged up to 3,000 from each ontology and performed random, independent sampling to ensure that the result is asymptotically correct.

# CHAPTER 6

# SUMMARY AND CONTRIBUTIONS

This dissertation is concerned with the general problem of how to assist users with searching for Web pages. One of the suggested solutions for this problem was utilizing ontologies that are constructed with natural concepts, IS-A relationships, and semantic relationships to improve the quality of Web search results. It was also suggested that information in the Deep Web should be exploited to build domain ontologies. Hence, major contributions of this dissertation are twofold. The first one is designing and implementing algorithms that extract useful information from the Deep Web to build ontologies and the other is quantifying the quality of ontologies (QoO) from the naturalness point of view.

In order to achieve the goals of the Semantic Deep Web, this dissertation claimed that it is necessary to add more semantics to Deep Web processing. The semantics and the Deep Web processing are reciprocal to each other through the use of ontologies in the view of this dissertation.

The aforementioned approaches to generating domain ontologies automatically were shown to provide not only improved processing of users' queries, but also better ontology usage for specifying Deep Web data sources. Web sites themselves provide an environment for users to learn domain specific terms, which implicitly reflect users' understanding of the domain. The necessity of automatically extracting frequently used concepts across many Deep Web sites was discussed and a sample was implemented because the domain ontologies exhibit an important feature, domain consensus which

makes the ontologies usable.

Contributions of the implemented approach to the semi-automatic enrichment of domain ontologies included the automatic extraction of instances for the domain ontology from rich Deep Web data sources and improved Deep Web Search. Semantics at the data level of ontologies became available by crawling data from the Deep Web. Automatic programs were developed for representing the semantics in the Web Ontology Language (OWL) to facilitate development of the Semantic Deep Web. The presented prototype of a Deep Web search system showed one aspect of how to utilize domain ontologies to meet Web users' needs, that is an automated process to interpret his information needs against the backdrop of the Deep Web.

This dissertation also attacked the problem of "naturalness" as an aspect of the QoO and developed the appropriate "measuring instruments" employing the Web as the corpus for the evaluation. An ontology maintenance model implied by this evaluation methodology, was designed such that an interactive process with the Web users' inputs will replace unnatural concepts and relationships by more natural ones, increasing the usability of ontologies for the Semantic Deep Web.

Extracting concepts, IS-A relationships, and semantic relationships was dealt with in Chapters 2, 3, and 4, respectively. Instances were also extracted from the Deep Web so as to demonstrate the effectiveness of the presented methodology in Chapter 4. Finally, an evaluation of the naturalness of ontologies as a QoO measure was presented in Chapter 5.

### 6.1 Automatic Attribute Extraction from the Deep Web

Chapter 2 presented a novel approach to automatic extraction of attributes from query interfaces of the Deep Web in order to address the current limitations in accessing Deep Web data sources. The Automatic Attribute Extraction (AAE) algorithm (1) identified attributes that are used by query Web page designers, called Programmer Viewpoint Attributes, and (2) attributes that are presented as labels to users, called User Viewpoint Attributes. By matching attributes used by designers and users with support of a synonym analysis based on an ontology, WordNet, the set of attributes, called the set of Final Attributes was extracted in eight e-commerce related domains. The size of the final attributes set from 477 Web data sources in eight domains (i.e., airfares, automobiles, books, car rentals, hotels, jobs, movies, and music records) is 1,825 which is about three times larger than the manually extracted attribute set. Utilizing WordNet in the Deep Web context, the Semantic Deep Web was newly introduced as a combination of aspects of the Deep Web and aspects of the Semantic Web.

The AAE algorithm was assessed by its outputs to evaluate whether the set of final attributes was comparable to the set of manually extracted attributes in locating Web data sources relevant to users' queries. The "manual attribute set" showed an average precision of 0.96 while the "automatic attribute set" showed an average precision of 0.74. Based on the results of comparison tests conducted, it was concluded that the AAE algorithm can work like an expert Web programmer in extracting attributes from the query interfaces of the Deep Web.

### 6.2 Automatic Generation of Ontology from the Deep Web

Chapter 3 presented the algorithm to automatically generated domain ontologies. The attributes extracted by the AAE in Chapter 2 served as concepts to be amalgamated to form domain ontologies. As a result, the eight domain ontologies were automatically generated where in all, 1825 concepts were interwoven into the IS-A relationships of WordNet and 5473 concepts directly from WordNet were included in the generating process. These domain ontologies were represented in the Web Ontology Language (OWL) to be used by both computers and humans throughout the Web.

One of the most significant aspects of the presented approach lies in enriching WordNet by domain specific information of the Deep Web sources. Another aspect was its automatic process for on-line generation of domain ontologies along with data extraction from Deep Web sources.

### 6.3 Instance Extraction for Ontology-based Deep Web Search

Chapter 4 extended the presented domain ontologies in Chapter 3 by including new concepts and instances, which may not exist in WordNet. A novel approach to enhancing a domain ontology was presented by adding 3385 instances of concepts and 8494 instances of semantic relationships extracted from a Deep Web site (http://www.united.com/) to it. Secondly, a prototype Web search system, which utilizes this domain ontology, was implemented.

The success of searching for Web pages was evaluated for flight-related Web sites, while using the domain ontology augmented with instances. Criteria for the successful search include $|W_{SE}| \lhd |W_S|$ and $|(M_W^{SE})_{100}| \rhd |(M_W^S)_{100}|$. Without the presented method,

users would have to review $|\overline{W_S}| = 134{,}954{,}000$ Web sites, which would take too long for users to find all relevant Web sites of interests. Thanks to the suggested method, users would need to review only $|\overline{W_{SE}}| = 18{,}317$ Web sites. This is still very large. However, the number of relevant Web sites in the first one hundred Web sites increased from $|\overline{(M_W^S)_{100}}| = 4$ to $|\overline{(M_W^{SE})_{100}}| = 14$ on average. Hence, the first criterion $|W_{SE}| \lessdot |W_S|$ implies huge savings of users' time and efforts and the second criterion, $|(M_W^{SE})_{100}| \gtrdot |(M_W^S)_{100}|$ implies the better quality of the search results.

In summary, the experiments suggested that Web search results returned by the search engine were improved by this new approach. More sites relevant to a user's needs could be located faster since users would need to examine fewer results and would find more relevant Webs sites early on in the result pages.

### 6.4 Ontology Evaluation: Naturalness Perspective

Chapter 5 presented the problem of ontology 'naturalness' by defining its meaning for the concepts of an ontology and the relationships between the concepts of an ontology in order to make the ontology (more) understandable to Semantic Web users. Chapter 5 analyzed several existing ontologies (WordNet, UMLS, etc.) with respect to the quality of 'naturalness' formalized as a quantitative measurement of the ontologies.

Evaluating naturalness of concepts of the ontologies produced the following results: (1) the UMLS Semantic Network is the most natural, followed by (2) WordNet and OpenCyc Class, (3) OpenCyc (Complete), (4) OpenCyc Individual, and (5) the UMLS Metathesaurus.

The comparative analysis of naturalness was extended to the structure of the ontologies: the naturalness of IS-A relationships and non IS-A relationships of concept pairs was measured. The least natural ontology (component) was the Metathesaurus while the Semantic Network was the best, but the number of concepts in it is comparatively very small, too small to be useful in practice. Thus, it was concluded that WordNet was the most natural large ontology to be used to define a likely upper limit or at least a reference point in evaluating naturalness of other large ontologies.

The property of naturalness was also applied in evaluating the eight domain ontologies generated in Chapter 3. As a result, naturalness values both for concepts and IS-A relationships was higher than for the UMLS, WordNet and OpenCyc. An average of naturalness of concepts in the eight domain ontoloies was 364,854,988, whereas the highest one (the UMLS Semantic Network) among existing large ontologies, was 5,059,901. Among the eight domain ontologies, the Airfares domain ontology had the lowest and it was 238,656,258 which is much higher than the highest one in the existing large ontologies. Thus, it was concluded that concepts in the domain ontologies are more natural than existing ontologies. Moreover, an average naturalness of IS-A relationships in domain ontologies was 103,067,369 while the UMLS Semantic Network had the highest value, 323,777 among existing large ontologies. Thus, it was also concluded statistically that IS-A relationships in the domain ontologies are more natural than existing ontologies.

# CHAPTER 7

# FUTURE WORK

Chapter 7 summarizes restrictions and limitations for methods presented in this dissertation with possible solutions. Several open problems in this dissertation are identified for future work as well.

## 7.1 Limitations and Restrictions

If readers or other researchers attempt to replicate the experiments conducted in this dissertation, they will face two major restrictions: disconnection by the Google server and blocking by Deep Web sites.

First, in order to assess the naturalness of ontologies in Chapter 5, the frequency information, called Google#, provided by Google was utilized. However, Google limits users to 1000 queries per day by their policy. By sending over a hundred thousand concepts and their relationships to the Google search engine, the remote server in the Google company may produce frequent disconnection errors and long processing times because of this limit. There are two possible solutions so as to mitigate this problem. Just like the evaluation of the existing large ontologies was conducted with special permission from the Google company (Section 5.4), one may ask Google for the increased maximum of 20,000 queries per day for research. Another solution is sampling, which was done when evaluating the eight domain ontologies in Section 5.5.

The statistical analysis in Chapter 5 requires the normality assumption for the t-test robustness. As variables did not follow a normal distribution, the results from the

Satterthwaite method were alternatively presented which gives t-statistics in an asymptotic way. According to the Satterthwaite method, if the sample size is large and sampling is performed randomly and independently, the result is asymptotically correct. In this dissertation, the sample size from each ontology was only 3,000 or less because the aforementioned restriction by the Google company. In the future, a full test or larger sampling is necessary to find more exact measurements for the quality of the eight automatically generated ontologies.

There are many other databases of Deep Web sites which the automatic crawling program cannot access, due to access restrictions. Session tracking by a Web site is a known barrier for a Web crawling program, as a site may use cookies to trace interactive progress of a client. In addition, JavaScript may arbitrarily change a data representation which is independent of the visual interface shown to a client. This is another barrier against automatic analysis programs. In fact, while the crawling program was tested, several problems were detected in reading HTML results from a number of Web sites. A message, "Please activate scripting" means that the site detects a Web browser of a client and its script option before it answers a user query. Thus, a crawling program needs a facility of a kind of Web browser. On the other hand, a message of "Permission denied" was encountered when the JavaScript program, which runs on a university server, directly reads result pages which were displayed on the screen (i.e., client side) but streamed from a company Website. This method of access is called *cross-site scripting.* It causes security breaches, and thus, is prohibited. Considering this dissertation's research result, namely that ontology instances extracted from the Deep Web contribute greatly to locating relevant Websites, community wide efforts are

necessary to extract large numbers of instances from the Deep Web. Only a collective approach can make the Semantic Deep Web a reality.

## 7.2 Open Problems

Amongst numerous possible extensions of this dissertation, six of them are identified here: domain ontology usage for semantic annotation of Web services, direct utilization of the Deep Web for ontology building, further semantic relationship utilization, ontology integration, further enrichment of the domain ontology with synonyms, and human subject tests.

Two intended advantages of the presented approach to generating a good domain ontology are:

1. It improves the processing of a users' queries.
2. It can be used for semantic annotation of Web services since mapping service descriptions to concepts of the ontologies is essential for discovery of the services.

The former one was demonstrated but the latter was not addressed in this dissertation. Demonstrating the domain ontology usage for semantic annotation of Web services remains open for future work.

IS-A relationtionships between attributes automatically generated from the Deep Web in Chapter 3 are only verified using WordNet. However, the Deep Web contains useful relationship information as well. Utilizing this information to build better ontologies remains an open problem.

In the future, more semantics needs to be added to Deep Web processing, to

achieve the goal of a Semantic Deep Web. For example, WordNet might be replaced or augmented by domain-specific ontologies. While WordNet has excellent wide coverage and usability it lacks domain specific knowledge. We have reviewed existing ontologies for E-commerce and have not found any ontology with both the breadth and depth needed for this project. Thus building such an ontology containing more semantic relationships is future work.

Ontology integration is another open problem. In this dissertation, eight domain ontologies were developed. These can be realized as sub domain ontologies for a larger domain ontology preferably called E-commerce. Integrating multiple sub-domain ontologies into one remains an open problem. Moreover, as a new Website is visited, possibly, new instances and concepts can be recognized and merged into existing ontologies. Such learning of ontology concepts is very important and is another open problem.

The domain ontologies constructed in this dissertation should be further enriched with synonyms. WordNet was used for taxonomic weaving of concepts. The generated ontologies may not support automatic semantic matching for a user query term that is too far away from a Web site term, based on a string comparison. For example, "Leaving From" and "City Name" would not be considered synonyms normally, even with flexible matching, although in our domain they describe analog fields, and are therefore synonymous. Hence, constructing an ontology which will be sufficiently flexible to recognize "City Name" as synonym of a query term "Departure City" is another open problem.

Finally, many claims in this dissertation could be better justified with human subject tests. Especially in Chapter 4, $|W_{SE}| \leq |W_S|$, $|M_W^{SE}| > |M_W^S|$, and $|(M_W^{SE})_{100}| > |(M_W^S)_{100}|$ were claimed. Albeit very limited human subject tests were conducted to validate $|W_{SE}| \leq |W_S|$ and $|(M_W^{SE})_{100}| > |(M_W^S)_{100}|$ for a small number of $|SE|$ vs. $|S|$ cases, further full human subject tests are necessary for more cases. Justifying $|M_W^{SE}| > |M_W^S|$ claim would require enormous efforts and time of a large group of human subjects. Moreover, empirical tests with human subjects are also necessary to justify whether the naturalness of ontologies is in fact corroborated by human perception.

## APPENDIX

## PARTIAL LISTS OF CONCEPT PAIRS AND SEMANTIC TYPE RELATIONS

The partial lists of concept pairs of a simple experimental control group are as follows where MCPGN indicates the mean of numbers of Google search results of concept pairs:

| **Data Source: OpenCyc** | |
|---|---|
| **Concept-pair in IS-A relationship: MCPGN = 31128.1** | **Two non-relevant concepts (i.e. control group): MCPGN = 1981.7** |
| "Purposeful Action","Voting" | "Purposeful Action","Scoping Relation" |
| "Natural Gas", "Fossil Fuel" | "Fossil Fuel","Bread" |
| "Food Service Organization","Bakery Store" | "Food Service Organization","Street Bike" |
| "Beach Layia", "Flowering Plant" | "Flowering Plant","Short Bone" |
| "Iodic Acid", "Electrolyte" | "Electrolyte","Whale" |
| "Performance Degradation Computer", "Indefinite Wait Computer Performance" | "Performance Degradation Computer", "Horror TV Show" |
| "Happiness", "Feeling Attribute" | "Feeling Attribute", "Serve Warrant" |
| "Facial Tissue", "Consumable Product" | "Consumable Product", "Ice Hockey Player" |
| "Health Care", "Taking Care Of Something" | "Taking Care Of Something", "Audible Sound" |
| "Fruitcake", "Cake" | "Cake", "Stop Sign" |

| **Data Source: WordNet** | |
|---|---|
| **Concept-pair in IS-A relationship: MCPGN = 23,067** | **Two non-relevant concepts: MCPGN = 180.2** |
| "cameraman", "photographer" | "photographer", "dressing_station" |
| "prickly lettuce","compass plant" | "compass plant", "social insurance" |
| "electrophorus", "wimshurst machine" | "wimshurst machine","common_axe" |
| "mantlepiece", "shelf" | "mantlepiece","citrous_fruit" |
| "ridiculer","humorist" | "ridiculer","neutralist" |
| "braiding","trimming" | "braiding","successfulness" |
| "tequila", "liquor" | "liquor","successfulness" |
| "piezometer", "measuring_instrument" | "measuring instrument","ground cover" |
| "chronic obstructive pulmonary disease", "pulmonary emphysema" | "pulmonary emphysema","electro acoustic transducer" |
| "lectureship", "position" | "position","goose_pimple" |

**"SemanticTypesRelations" extracted from *srstr* in UMLS Semantic Network for our analysis**

| | | |
|---|---|---|
| adjacent_to | derivative_of | method_of |
| affects | developmental_form_of | occurs_in |
| analyzes | diagnoses | part_of |
| assesses_effect_of | disrupts | performs |
| associated_with | evaluation_of | practices |
| branch_of | exhibits | precedes |
| carries_out | indicates | prevents |
| causes | ingredient_of | process_of |
| co-occurs_with | interacts_with | produces |
| complicates | interconnects | property_of |
| conceptual_part_of | issue_in | result_of |
| conceptually_related_to | location_of | surrounds |
| connected_to | manages | traverses |
| consists_of | manifestation_of | treats |
| contains | measurement_of | tributary_of |
| degree_of | measures | uses |

**"SemanticTypesRelations" extracted from *mrrel* in UMLS Meta Thesaurus for our analysis**

| | | |
|---|---|---|
| access_instrument_of | has_episodicity | indirect_procedure_site_of |
| access_of | has_evaluation | induced_by |
| active_ingredient_of | has_expanded_form | induces |
| actual_outcome_of | has_expected_outcome | ingredient_of |
| adjectival_form_of | has_finding_context | intent_of |
| adjustment_of | has_finding_site | interpretation_of |
| affected_by | has_focus | interprets |
| affects | has_form | is_interpreted_by |
| analyzed_by | has_indirect_device | laterality_of |
| analyzes | has_indirect_morphology | location_of |
| approach_of | has_indirect_procedure_site | manifestation_of |
| associated_disease | has_ingredient | mapped_from |
| associated_finding_of | has_intent | may_be_diagnosed_by |
| associated_genetic_condit | has_interpretation | may_be_prevented_by |
| associated_morphology_of | has_laterality | may_be_treated_by |
| associated_procedure_of | has_location | may_diagnose |
| associated_with | has_manifestation | may_prevent |
| branch_of | has_measurement_method | may_treat |
| british_form_of | has_mechanism_of_action | measured_by |
| causative_agent_of | has_member | measurement_method_of |
| cause_of | has_method | measures |
| challenge_of | has_occurrence | mechanism_of_action_of |
| classified_as | has_onset | member_of_cluster |
| classifies | has_part | metabolic_site_of |
| clinically_associated_wit | has_pathological_process | metabolized_by |
| clinically_similar | has_pharmacokinetics | metabolizes |
| co-occurs_with | has_physiologic_effect | method_of |
| component_of | has_plain_text_form | modified_by |
| conceptual_part_of | has_precise_ingredient | modifies |
| consists_of | has_priority | multiply_mapped_from |
| constitutes | has_procedure_context | multiply_mapped_to |
| contained_in | has_procedure_device | noun_form_of |
| contains | has_procedure_morphology | occurs_after |
| contraindicated_with | has_occurrence | occurs_before |
| course_of | has_onset | occurs_in |
| ddx | has_part | onset_of |
| default_mapped_from | has_pathological_process | other_mapped_from |
| default_mapped_to | has_pharmacokinetics | other_mapped_to |
| definitional_manifestatio | has_physiologic_effect | part_of |
| degree_of | has_plain_text_form | pathological_process_of |
| diagnosed_by | has_precise_ingredient | pharmacokinetics_of |
| diagnoses | has_priority | physiologic_effect_of |
| direct_device_of | has_procedure_context | plain_text_form_of |
| direct_morphology_of | has_procedure_device | precise_ingredient_of |
| direct_procedure_site_of | has_procedure_morphology | primary_mapped_from |
| direct_substance_of | has_procedure_site | primary_mapped_to |
| divisor_of | has_process | priority_of |
| dose_form_of | has_property | procedure_context_of |
| drug_contraindicated_for | has_indirect_procedure_site | procedure_device_of |
| due_to | has_ingredient | procedure_morphology_of |
| encoded_by_gene | has_intent | procedure_site_of |
| encodes_gene_product | has_interpretation | process_of |
| episodicity_of | has_laterality | property_of |
| evaluation_of | has_location | recipient_category_of |
| exhibited_by | has_manifestation | result_of |
| exhibits | has_measurement_method | revision_status_of |
| expanded_form_of | has_mechanism_of_action | scale_of |
| expected_outcome_of | has_member | scale_type_of |
| finding_context_of | has_method | severity_of |

| | | |
|---|---|---|
| finding_site_of | has_recipient_category | sib_in_branch_of |
| focus_of | has_result | sib_in_tributary_of |
| form_of | has_revision_status | site_of_metabolism |
| has_access | has_scale | specimen_of |
| has_access_instrument | has_scale_type | specimen_procedure_of |
| has_active_ingredient | has_severity | specimen_source_identity_of |
| has_actual_outcome | has_specimen | specimen_source_morphology_of |
| has_adjustment | has_specimen_procedure | specimen_source_topography_of |
| has_approach | has_specimen_source_identity | specimen_substance_of |
| has_associated_finding | has_specimen_source_morphology | ssc |
| has_associated_morphology | has_specimen_source_topography | subject_relationship_context_of |
| has_associated_procedure | has_specimen_substance | suffix_of |
| has_branch | has_subject_relationship_context | supersystem_of |
| has_british_form | has_suffix | system_of |
| has_causative_agent | has_supersystem | temporal_context_of |
| has_challenge | has_system | time_aspect_of |
| has_component | has_temporal_context | tradename_of |
| has_conceptual_part | has_time_aspect | translation_of |
| has_contraindicated_drug | has_tradename | treated_by |
| has_contraindication | has_ translation | treats |
| has_course | has_tributary | tributary_of |
| has_definitional_manifestation | has_version | uniquely_mapped_from |
| has_degree | has_xml_form | uniquely_mapped_to |
| has_direct_device | icd_asterisk | use |
| has_direct_morphology | icd_dagger | used_by |
| has_direct_procedure_site | indicated_by | used_for |
| has_direct_substance | indicates | uses |
| has_divisor | indirect_device_of | version_of |
| has_dose_form | indirect_morphology_of | xml_form_of |

## REFERENCES


1. Agirre, E., Ansa, O., Hovy, E., and Martinez, D. (Aug. 2000). Enriching very large ontologies using the WWW. In *Proceedings of ECAI Workshop on Ontology Learning*, 37-42.

2. Alani, H., and Brewster, C. (2005). Ontology Ranking based on the Analysis of Concept Structure. In *Proceedings of the Third International Conference on Knowledge Capture (K-Cap)*, 51-58.

3. An, Y. J., Huang, K.-C., and Geller, J. (2006) Naturalness of Ontology Concepts for Rating Aspects of the Semantic Web. In *Communications of the International Information Management Associatio*n, 6 (3), 63-76.

4. An, Y. J., Geller, J., Wu, Y.-T., and Chun, S. A. (2007a). Semantic Deep Web: Automatic Attribute Extraction from the Deep Web Data Sources. *Proceedings of the 22nd Annual ACM Symposium on Applied Computing (SAC* 2007), 1667-1672.

5. An, Y. J., Huang, K.-C., and Geller, J. (2007b). Rating the Naturalness of Ontology Taxonomies. In *Proceedings of the 20th International FLAIRS Conference*, 176-177.

6. An, Y. J., Geller, J., Wu, Y.-T. and Chun, S. A. (2007c). Automatic Generation of Ontology from the Deep Web. In *Proceedings of 6th International Workshop on Web Semantics (WebS*07), 470-474.

7. An, Y. J., Huang, K.-C., and Geller, J. (2007d). Comparative Anatomy of Ontologies: the Semantic Naturalness Perspective. Submitted for Publication.

8. Arasu, A., and Garcia-Molina, H. (June 2003). Extracting structured data from web pages. In *Proceedings of the ACM SIGMOD Conference*, 337-348.

9. Bechhofer, S., Harmelen, F. V., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F.,and Stein, L. A. (Feb 2004). OWL Web Ontology Language Reference Retrieved April 2005 from The World Wide Web Consortium Web site : http://www.w3.org/TR/owl-ref/

10. Bergman, M. K. (2006). The Deep Web: Surfacing Hidden Value. Retrieved May 2006 from http://www.brightplanet.com/technology/DeepWeb.asp

11. Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5),34-43.

12. Bjelogrlic, Z., Van Gulik, D. W., Reggiori, A., and Norheim, D. (2006). Semantic Web: A New Dimension in Distributed Computing. Retrieved May 2006 from http://www.asemantics.com/presos/eusc2004/

13. Brewster, C., Ciravegna, F., and Wilks, Y. (2003). Background and Foreground Knowledge in Dynamic Ontology Construction: Viewing Text as Knowledge Maintenance. In *Proceedings of the Semantic Web Workshop* (*SIGIR*), Toronto, Canada.

14. Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. Retrieved June 13, 2006, from Stanford University, InfoLab Web site: http://www-db.stanford.edu/~backrub/google.html.

15. Castano, S., and Antonellis, V. D. (1999). A Discovery-Based Approach to Database Ontology Design. *Distributed and Parallel Databases - Special Issue on Ontologies and Databases*, 7(1), 67-98.

16. Chang, K. C., He, B., Li, C., and Zhang, Z. (2003). The UIUC Web Integration Repository. Retrieved from MetaQuerier: Exploring and Integrating the Deep Web Web site: http://metaquerier.cs.uiuc.edu/repository/

17. Chang, K. C., He, B., Li, C., Patel, M., and Zhang, Z. (2004). Structured Databases on the Web: Observations and Implications. SIGMOD Record, 33(3), 61-70.

18. Chun, S. A. and Geller, J. (2008). Evaluating Ontologies based on the Naturalness of their Preferred Terms. In *Proceedings of the 41st Hawaii International Conference on System Sciences*.

19. Colomb, R. M. (2002). Quality of Ontologies in Interoperating Information Systems. Retrieved June 13, 2006, from the Institute of Cognitive Science and Technology, Laboratory for Applied Ontology Web site: http://www.loa-cnr.it/Papers/ISIB-CNR-TR-18-02.pdf

20. Cycorp (2002). Rule Macro Predicates Retrieved Aug 08, 2006, from http://www.cyc.com/cycdoc/vocab/rule-macro-vocab.html

21. Cycorp (2005). OpenCyc (Version 0.9) [Computer program]. Retrieved June 16, 2006, from http://www.opencyc.org/releases/

22. Cycorp (2006). Cyc 101 Tutorial. Retrieved June 16, 2006, from http://www.opencyc.org/doc/tut/?expand_all=1

23. Davulcu, H., Vadrevu, S., Nagarajan, S., and Ramakrishnan, I. V. (2003). OntoMiner: bootstrapping and populating ontologies from domain-specific Web sites. *IEEE Intelligent Systems*, 18(5), 24-33.

24. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J. (2004). Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, 652-659.

25. Doan, A., Madhavan, J. , Dhamankar, R., Domingos, P., and Halevy, A. (Nov. 2003). Learning to Match Ontologies on the Semantic Web. *The VLDB Journal*, 12(4), 303-319.

26. Dou, D., McDermott, D. V., and Qi P. (2005). Ontology Translation on the Semantic Web. *Journal on Data Semantics*, 2, 35-57.

27. Fensel, D., Hendler, J., Liebeman, H., Wahister, W. (2003). Spinning the Semantic Web. *Bringing the World Wide Web to Its Full Potential*, Cambridge, MIT Press, 1-25.

28. Florescu, D., Levy, A., and Mendelzon, A. (1998). Database techniques for the world-wide web: A survey. *ACM SIGMOD Record*, 27(3), 59-74.

29. Gal, A., Modica, G., Jamil, H. and Eyal, A. (2005). Automatic Ontology Matching using Application Semantics. *AI Magazine*, 26 (1), 21-31.

30. Garofalakis, M. N., Gionis, A., Rastogi, R., Seshadri, S., Shim, K. (2000). XTRACT: A System for Extracting Document Type Descriptors from XML Documents. In *Proceedings of Association for Computing Machinery/ Special Interest Group on Management of Data (ACM SIGMOD)*, 165-176.

31. Ghanem, T. M. and Aref, W. G. (2004). Databases deepen the Web. *Computer*, 37(1), 116-117.

32. Gligorov, R., Aleksovski, Z., Kate, W. and Harmelen, F. (2007). Using Google Distance to Weight Approximate Ontology Matches. In *Proceedings of the 16th International World Wide Web Conference*, 767-775.

33. Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *Proceedings of International Workshop on Formal Ontology*, Retrieved from Stanford Univ. Knowledge Sharing Effort public library Web site: http://www-ksl.stanford.edu/knowledge-sharing/papers/README.html#onto-design

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

35. Handschuh, S., Volz, R. and Staab, S. (2003). Annotation for the Deep Web. *IEEE Intelligent Systems*, 18(5), 42-48.

36. Hawking, D., Craswell, N., Thistlewaite, P. B. and Harman, D. (1999). Results and Challenges in Web Search Evaluation. *Computer Networks* 31(11-16),1321-1330.

37. Hawking, S. W. (2002). On the Shoulders of Giants: The Great Works of Physics and Astronomy, Running Press, 391-399.

38. Hearst, M. A. (1992). Automatic Acquisition of Hyponyms form Large Text Corpora. In *Proceedings of COLING-92*, 539-545.

39. He, B., Patel, M., Zhang, Z. and Chang, K. C.-C. (2007). Accessing the Deep Web: A Survey. *Communications of the ACM* (CACM), 50(5), 94-101.

40. Heß, A., and Kushmerick., N. (2004). Machine learning for annotating semantic web services. In *Proceedings of AAAI 2004 Spring Symposium on Semantic Web Services*, 577-583.

41. Hieu, Q. L. (2005). Integration of Web Data Sources: A Survey of Existing Problems, In *Proceedings of the 17th GI-Workshop on the Foundations of Databases*, 78-82.

42. Ipeirotis, P. (2004). Classifying and Searching Hidden-Web Text Databases. Ph.D. Dissertation, Columbia University (advisor: L. Gravano).

43. Jansen, Bernard J.; Spink, A.; Saracevic, T. (2000). Real Life, Real Users, and Real Needs: A Study and Analysis of User Queries on the Web. *Information Processing & Management*, 36(2), 207-227.

44. Jansen, B. J. and Molina, P. (2006) The Effectiveness of Web Search Engines for Retrieving Relevant Ecommerce Links. *Information Processing & Management*, 42(4), 1075-1098

45. Kabra, G., Li, C., and Chang, K. C. (2005). Query Routing: Finding Ways in the Maze of the DeepWeb. In *Proceedings of the International Workshop on challenges in Web Information Retrieval and Integration*, 64-73.

46. Kalfoglou, Y. and Hu, B. (2006). Issues with evaluating and using publicly available ontologies. In *Proceedings of the Fourth International Evaluation of Ontologies for the Web Workshop* (EON2006), Retrieved from ECON2006 Web site: http://km.aifb.uni-karlsruhe.de/ws/eon2006/eon2006kalfoglouetal.pdf

47. Leacock, C., Chodorow, M., and Miller, G. A. (1998). Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 24(2).

48. Lee, Y., Geller, J. (2005). Semantic Enrichment for Medical Ontologies. *Journal of Biomedical Informatics*, 39(2), 209-226.

49. Lewis, D. (1983). New Work for a Theory of Universals. *Australasian Journal of Philosophy*, 61, 343-377.

50. Liddle, S., Embley, D., Scott, D. and Yau, S. (2002). Extracting Data Behind Web. In *Proceedings of the Joint Workshop on Conceptual Modeling Approaches for E-business: A Web Service Perspective* (eCOMO 2002), 38-49.

51. Lister Hill National Center for Biomedical Communications, U.S. National Library of Medicine (Aug. 2005). UMLS Knowledge Source Server. Retrieved August 08, 2006 from http://umlsks.nlm.nih.gov/kss/servlet/Turbine/template/admin,user,KSS_login.vm.

52. Lister Hill National Center for Biomedical Communications, U.S. National Library of Medicine (2006). The UMLS Semantic Network. Retrieved June 13, 2006, from U.S. National Institutes of Health Web site: http://semanticnetwork.nlm.nih.gov/

53. Maedche, A., and Staab, S. (2000). Mining Ontologies from Text. In *Proceedings of EKAW-2000*, Springer Lecture Notes in Artificial Intelligence (LNAI-1937), Juan-Les-Pins, France.

54. McCray, A. T., Burgun, A. and Bodenreider, O. (2001). Aggregating UMLS Semantic Types for Reducing Conceptual Complexity. In *Proceedings of Medinfo*, 171-175.

55. McDowell, L. K., and Cafarella, M. (2006). Ontology-driven Information Extraction with OntoSyphon. In *Proceedings of Internal Semantic Web Conference* (ISWC'06), 428-444.

56. Mihalcea, R., and Moldovan, D. I. (1999). An Automatic Method for Generating Sense Tagged Corpora. In *Proceedings of American Association for Artificial Intelligence*, Orlando, FL, 461-466.

57. Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11), 39-41.

58. Missikoff, M. Navigli, R. Velardi, P. (2002). The Usable Ontology: An Environment for Building and Assessing a Domain Ontology. In *Proceedings of the First Int. Semantic Web Conference*, 39-53.

59. Mizoguchi, R., and Ikeda, M. (1996). Towards Ontology Engineering. *Technical Report AI-TR-96-1*, The Institute of Scientific and Industrial Research, Osaka University.

60. Mizoguchi, R. (2004). Tutorial on Ontological Engineering: Part 3: Advanced Course of Ontological Engineering. *New Generation Computing*, 22(2).

61. Modica, G., Gal, A., and Jamil, H. M. (2001). The Use of Machine-Generated Ontologies in Dynamic Information Seeking. In *Proceedings of the Sixth International Conference on Cooperative Information Systems* (CoopIS 2001), 433-447.

62. Navigli, R. (2002). Automatically extending, pruning, and trimming general purpose ontologies. In *Proceedings of the 2nd IEEE Conf on Sys Man Cybernetics*, 631-635.

63. Necib, C. B., and Freytag, J. (2003). Ontology Based Query Processing in Database Management Systems. In *Proceedings of the 6th International Conference on Ontologies, Databases and Applications of Semantics for Large Scale Information Systems* (ODBASE'2003), 37-99.

64. Nestorov, S., Abiteboul, S., and Motwani, R. (1998). Extracting schema from semistructured data. *ACM SIGMOD Record*, 27(2), 295-306.

65. Noy, N. F. and Musen, M. A. (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of AAAI-2000*, Austin, Texas, 450-455.

66. Ntoulas, A., Zerfos, P. and Cho, J. (2005). Downloading Textual Hidden Web Content Through Keyword Queries. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries* (JCDL), 100-109.

67. Obrst, L., Hughes, T. and Steve Ray, S. (2006). Prospects and Possibilities for Ontology Evaluation: The View from NCOR. In *Proceedings of the Fourth International Evaluation of Ontologies for the Web Workshop* (EON2006), Retrieved from ECON2006 Web site: http://km.aifb.unikarlsruhe.de/ws/eon2006/eon2006obrstetal.pdf.

68. Omelayenko, B. (2001) Learning of ontologies for the Web: the analysis of existent approaches. In *Proceedings of the International Workshop on Web Dynamics*, Retrieved from http://www.dcs.bbk.ac.uk/webDyn/webDynPapers/omelayenko.pdf.

69. Open Source Technology Group (2006a). OpenCyc (Version 0.9.5.Windows NT/2K/XP) [Computer program]. Retrieved June 16, 2006, from http://sourceforge.net/project/showfiles.php?group_id=27274.

70. Open Source Technology Group (2006b). Java WordNet Library (JWNL 1.3) [Computer program]. Retrieved June 16, 2006, from http://sourceforge.net/projects/jwordnet.

71. Patel-Schneider, P. F. (2004). What Is OWL (and Why Should I Care)? In *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning*, 735-737.

72. Patil, A., Oundhakar, S., Sheth, A. and Verma, K. (2004). METEOR-S: Web service annotation framework. Proceedings of the International World Wide Web (WWW), 553-562.

73. Pinto, H. S., and Martins, J. P. (2004). Ontologies: How Can They Be Built? *Knowledge and Information Systems*, 6(4), 441-464.

74. Pisanelli, D. M., Gangemi, A., and Steve, G. (1998). An Ontological Analysis of the UMLS Metathesaurus. *Journal of American Medical Informatics Association*, 5, 810-814.

75. Princeton University, Princeton, Cognitive Science Laboratory (2006). WordNet 2.1 Database Statistics. Retrieved June 13, 2006, from http://wordnet.princeton.edu/man/wnstats.7WN

76. Princeton University, Princeton, Cognitive Science Laboratory (2005). WordNet (Version 2.0) [Computer program]. Retrieved June 13, 2006, from http://wordnet.princeton.edu/oldversions

77. Raghavan, S., and Garcia-Molina, H. (2001). Crawling the Hidden Web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, 29-138.

78. Roitman, H., and Gal, A. (2006). OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources Using Sequence Semantics. In *Proceedings of International Conference on Extending Database Technology* (EDBT) Workshops, 573-576.

79. Sabou, M., Wroe, C., Goble, C. and Mishne, G. (2005). Learning domain ontologies for web service descriptions: an experiment in bioinformatics. In *Proceedings of the International World Wide Web* (WWW), 190-198

80. Seta, K., Ikeda, M., Kakusho, O., Mizoguchi, R. (1997). Capturing a Conceptual Model for End-User Programming: Task Ontology As a Static User Model. In *Proceedings of the Sixth International Conference on User Modeling* (UM'97), Chia Laguna, Sardinia, Italy, 203-214.

81. Singh, M. P. (2002). Deep Web structure. *IEEE Internet Computing*, 6(5), 4-5.

82. Smith, B.C. (1982). Reflection and Semantics in a Procedural Language, PhD thesis. Massachusetts Institute of Technology. MIT-LCS-272:154, Retrieved from MIT Web site: http://www.lcs.mit.edu/publications/specpub.php?id=840

83. Staab, S. and Maedche, A. (2000). Axioms are objects too - Ontology engineering beyond the modeling of concepts and relations, Research report 399, University of Karlsruhe, Institute AIFB.

84. Stanford Medical Informatics (2007a). Protégé 3.2.5 [Computer program API], Retrieved May, 2007, from http://protege.stanford.edu/doc/pdk/api/index.html

85. Stanford Medical Informatics (2007b). Protégé - OWL 3.2.1 [Computer program API], Retrieved May, 2007, from http://protege.stanford.edu/download/release-javadoc-owl/.

86. Stanford Center for Biomedical Informatics Research (2007). Protégé 3.3.1 [Computer Program], from http://protege.stanford.edu/download/registered.html.

87. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S. (2001). Ontology-Based Integration of Information - A Survey of Existing Approaches. In *Proceedings of the International Joint Conference on*

*Artificial Intelligence* (IJCAI-01) Workshop: Ontologies and Information Sharing, 108-117.

88. Weber, N. and Buitelaar, P. (2006). Web-based Ontology Learning with ISOLDE. In *Proceedings of ISWC2006* Workshop on Web Content Mining with Human Language Technologies, Retrieved from http://www2.dfki.de/~paulb/ISWC06.WebContentMining.pdf.

89. World Wide Web Consortium (1999) HTML 4.01 Specification Retrieved May 2006 from W3C. Technical Reports and Publications Recommendation Web site: http://www.w3.org/TR/html4/.

90. World Wide Web Consortium (2004). OWL Web Ontology Language Overview, Retrieved May, 2007 from http://www.w3.org/TR/owl-features/.

91. UC Berkeley (2006). Invisible or Deep Web: What it is, Why it exists, How to find it, and Its inherent ambiguity, Retrieved from UC Berkeley Library Web site: http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/InvisibleWeb.html.

92. University of California, Los Angeles, Academic Technology Services (2006). SAS Annotated Output. Retrieved June 16, 2006, from http://www.ats.ucla.edu/stat/sas/output/ttest.htm

93. U. S. National Library of Medicine (2003). Fact Sheets: UMLS Metathesaurus. Retrieved August 08, 2006 from http://www.nlm.nih.gov/archive/20031120/pubs/factsheets/umlsmeta.html.

94. U. S. National Library of Medicine ( 2005). UMLSKS API [Computer program]. Retrieved April 21, 2006 from http://umlsks.nlm.nih.gov/kss/servlet/Turbine/template/docs,api,apiDownload.vm; kss3#download.

95. U. S. National Library of Medicine (2006). About the UMLS® Resources. Retrieved from Unified Medical Language System Web site: http://www.nlm.nih.gov/research/umls/about_umls.html.

96. U. S. National Library of Medicine (2007a). UMLS Knowledge Sources. Retrieved from Unified Medical Language System Web site: http://www.nlm.nih.gov/research/umls/umlsdoc.html.

97. U.S. National Library of Medicine (2007b) UMLS® Release Notes and Problems (Bugs) 2007AC - Current UMLS Release. Retrieved from http://www.nlm.nih.gov/research/umls/release_notes.html.

98. Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E. G. M., and Milios, E. E. (Nov. 2005). Semantic similarity methods in WordNet and their application to information retrieval on the web. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, 10-16.

99. Wu, W., Doan, A., Yu, C., and Meng, W. (2005). Bootstrapping Domain Ontology for Semantic Web Services from Source Web Sites. In *Proceedings of the VLDB workshop on Technologies for E-Services*, 11-12.

100. Wu, W., Doan, A., Yu, C. (2005). Merging Interface Schemas on the Deep Web via Clustering Aggregation. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 801-804.

101. Zeng, Q. T., Tse, T., Crowell, J., Divita, G., Roth, L. and Browne, A. C. (2005) Identifying Consumer-Friendly Display (CFD) Names for Health Concepts, In *Proceedings of the AMIA Annual Symposium*, 859-863.