An Ontology-based Application to Detect, Annotate and Search web documents: First results

Jawad Bin Husain

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

April 2004

Your file Votre référence
ISBN: 0-612-91045-8
Our file Notre référence
ISBN: 0-612-91045-8

ABSTRACT

An Ontolgy-based Application to Detect, Annotate
and Search web documents: First results

Jawad Bin Husain

Ontologies are going to play a significant role in the future semantic web. They help organize the information about domains to facilitate not only to retrieve information about the domain but also reasoning and derivation of new information which can be used by search engines and software agents. This major report discusses the resources offered by ontologies and explores the way to use them in web document detection, annotation and search. Since an ontology captures all the significant terms and their hierarchical relationships for a specific domain, it motivates us to use this information to detect web documents which fall under that domain. The content of the web documents can be analyzed to find out the number of ontology terms encountered in those documents and accordingly assign them rank indicating how close they are to the corresponding domain. Once we know the encountered ontology terms in the web document, we can annotate the terms with appropriate web links where more information about the terms can be found. The domain detection approach can also be very useful in searching web documents. If we know the search domain in addition to search terms, the search can be more effective and accurate. While searching the web, we usually get search results containing documents from different domains whereas we are interested in the documents of a particular domain. Domain detection technique using ontologies might solve this problem. In this report, the algorithm of domain detection, annotation and search for web documents exploiting the resources offered by ontologies has been explained and implemented as a Java application. The report also discusses the design of a test ontology that has been used to test the implemented system.

TABLE OF CONTENTS

List of Figures

1. Introduction

The goal of this major report is to explore the applications of ontologies in the field of web document domain detection, annotation and search. It is an effort to exploit the resource or information about the domains provided by the corresponding ontologies. In this report, first the definition and need of ontologies are discussed. Then the motivations of using ontologies in the area of domain detection, annotation and search for web documents are explained. Based on the motivation, the problem is defined, the problem solving approach is formulated and subsequently the algorithm is proposed. Finally a Java application is developed to implement the algorithm. As a test domain, university faculty home page has been selected. An ontology representing university faculty home page has been designed and used in the experiments to test the implementation. In the result section, the implemented system is used to process several sample web documents to demonstrate its ability in domain detection, annotation and in improving web search accuracy.

2. Background

2.1 What is an ontology?

"An ontology is a formal explicit description of concepts in a domain of discourse (**classes** (sometimes called **concepts**)), properties of each concept describing various features and attributes of the concept (**slots** (sometimes called **roles** or **properties**)), and restrictions on slots (**facets** (sometimes called **role restrictions**)). An ontology together with a set of individual **instances** of classes constitutes a **knowledge base**. In reality, there is a fine line where the ontology ends and the knowledge base begins. Classes are the focus of most ontologies. Classes describe concepts in the domain. For example, a class of wines represents all wines. Specific wines are instances of this class. A particular Bordeaux wine is an instance of the class of Bordeaux wines. A class can have **subclasses** that represent concepts that are more specific than the superclass. For example, we can divide the class of all wines into red and white wines or into sparkling and non-sparkling wines." [1]

2.2 Why develop an ontology?

"Ontologies have now become more common in the World-Wide Web. The ontologies in the Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (such as on Amazon.com). The WWW Consortium (W3C) has already declared OWL (Web Ontology Language) as recommended standard to be used to publish and share ontologies, supporting advanced

Web search, software agents and knowledge management. [4] Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED and the semantic network of the Unified Medical Language System. Broad general-purpose ontologies are emerging as well. An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

Some of the important reasons for developing ontologies are:

- To share common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To separate domain knowledge from the operational knowledge.

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

Enabling reuse of domain knowledge was one of the driving forces behind the recent surge in ontology research. For example, models for many different domains need to represent the notion of time. This representation includes the notions of time intervals,

points in time, relative measures of time, and so on. If one group of researchers develops such an ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain. We can also reuse a general ontology and extend it to describe our domain of interest.

Separating the domain knowledge from the operational knowledge is another common use of ontologies. We can describe a task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves. For example, we can develop an ontology of PC-components and characteristics and then develop an algorithm to configure made-to-order PCs that uses that ontology. It is possible to develop the algorithm independent of the PC ontology so that the same algorithm can be used to configure elevators if we feed an elevator component ontology to it." [1]

3. Motivation

Since we are moving towards the semantic web, more and more ontologies will be available to represent domains in the web that can be easily processed by applications or agents to derive important information. This report is an effort to explore the way to take advantage of the resources offered by ontologies.

An ontology captures all the necessary concepts or terms of a domain. Each concept is represented as a class. More specialized concepts or classes are derived as children from more generalized parent concepts or classes to form a hierarchical structure of classes. So an ontology is a repository of all possible terms for its domain. The collection of terms can be a vital resource for application in web document domain detection, annotation and search. It will be even more useful if we can process the ontology to derive a mapping table storing all possible terms of the corresponding domain with some additional information for each term as follows:

- All possible uses and variations in English text for the term.
- Weight indicating how important or significant it is in determining the domain.
- The pointer, reference or URL that explains what it is and how to use or process it.

3.1 Motivation for document detection/annotation

The mapping table containing the ontology terms and their various uses can be very useful in determining how closely a web document relates to the given domain. In most

cases, the more the number of different ontology terms from the mapping table encountered in an arbitrary document, the more is the probability that the document falls under the domain. We can calculate a rank based on the total number of different terms encountered in the document so that a higher value of the rank will indicate higher degree of similarity of document to the domain. The length of the document is not taken into consideration for rank calculation that is further discussed in the future extension section.

Also since the mapping table can store the URL for the term definition and process information, we can annotate the ontology terms encountered in the document with the appropriate links.

One may argue that we can construct a mapping table without the use of an ontology. So why should we need an ontology? Yes, that is possible. But the point is to use the resources already provided by ontologies to save time and energy. Regarding this argument some points can be made as follows:

1. Ontologies will be a standard for representing domain knowledge over the web in the future. The class names or terms used in them will be standardized and agreed on for many domains. So the goal is to make use of this valuable resource.
2. Once an ontology is designed, implemented and available it can be used right away by generating the mapping table that will be used by the domain detection and annotation application. It will save a lot of time and energy that would have been wasted otherwise if the mapping table had to be constructed from scratch.

3.2 Motivation for a search module

Document domains can play a very important and effective role in searching the web. The knowledge about domains makes the search more accurate. For example, a user may want to search the word "admission" but may want to retrieve information only related to university admission but not hospital admission or admission used in any other domain. But the search engine will present the user with all the documents related to admission irrespective of the intended domain and the user will have to filter out the unnecessary documents manually. This is certainly a waste of time and energy. Another example is when a user wants to search the term "play". Some may be interested in sports, while others may be interested in cultural shows such as drama or opera. So it is important to know the domain a user is interested in. Therefore, it will be more effective and accurate if the search engine had the knowledge of both the search topic and the domain. Given that a domain is important in accurate search, the motivation here is that we can use the domain detection module discussed earlier to find only the documents in the search result falling under the given domain of user choice. It can be a 2-step process as follows:

1. Search for all the web documents where the search topic has been encountered at least once.
2. Pass all the documents in the search result through the domain detection process to find only the documents falling under the given domain.

The assumption that domains play an important role in document search can be supported by the following web search example.

3.2.1 Web search example

For our test case, the "university faculty home page" is selected as domain. Let us search for all university faculty home pages containing the search topic "publications". So, the goal of a user is to browse and explore through the publications located only in the home pages of university teachers. Most likely the user will search a common term such as "Publications". Using Google that is one of the best search engines available today, the findings are as follows (see figure 1):

1. Among the top-ranked 10 documents in the search result, there is no document that contains information about publications of any university faculty.
2. Also among the top-ranked 20 documents, no document says anything about publications of any university faculty.

Even if the user wants to fine-tune the search phrase using "university publications", "university teacher publications" etc., the search result will contain different university links, a few university publication sites and other unrelated sites but it does not contain any link directly to the publications of any university teacher's home page.

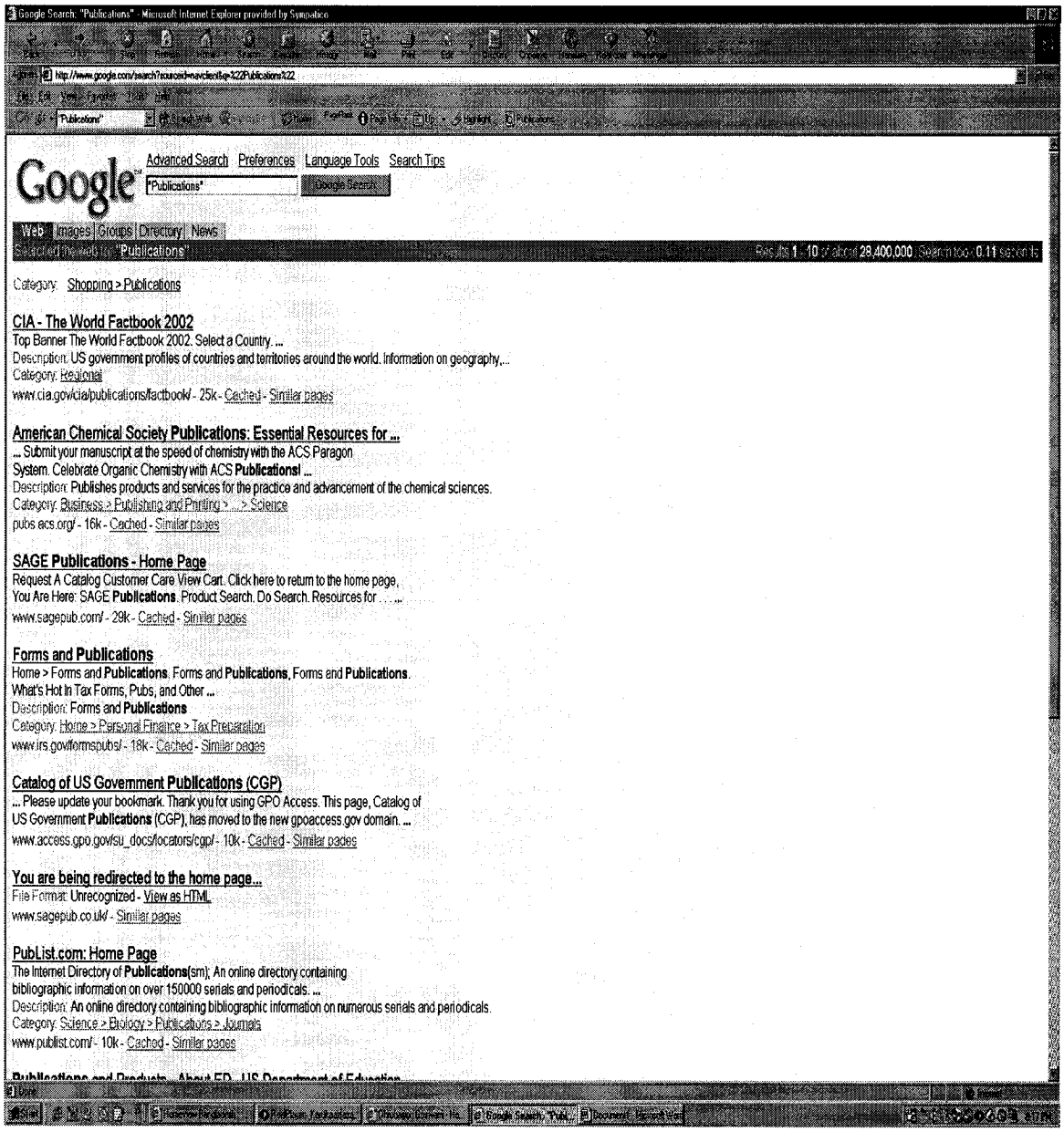

Figure 1: First page of the search result produced by Google for search topic "Publications"

In the same process, if any user wants to browse the homes pages of university faculties to learn about the courses offered by them, he or she would most likely search the term "Courses". Using Google, the snapshot of the search result is given below (see figure 2). The findings are:

1. Among the top-ranked 10 documents in the search result, there is no document that contains information about courses of any university faculty.
2. Also among the top-ranked 20 documents, nothing says anything about courses offered by any university faculty.

Using the search phrases are: "Current courses", "University teacher courses", "Courses offered by university teacher", the search will be more effective to find courses offered by different universities but no search will result in listing home pages of university teachers containing information about courses offered by them.

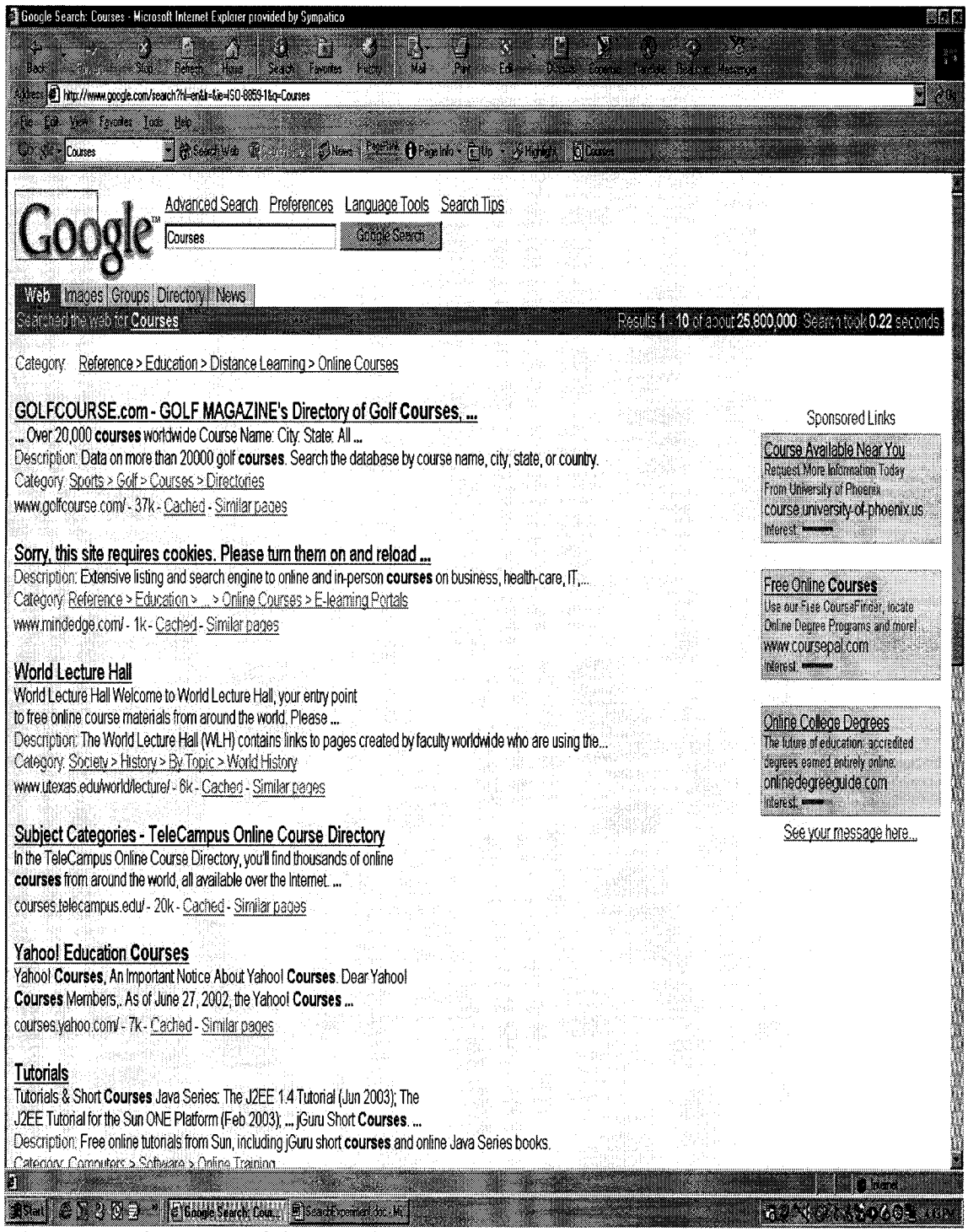

Figure 2: First page of the search result produced by Google for search topic "Courses"

For the terms: "Teachings" and "Research interests" we get the similar findings as illustrated by the following snapshots (see figures 3 and 4):

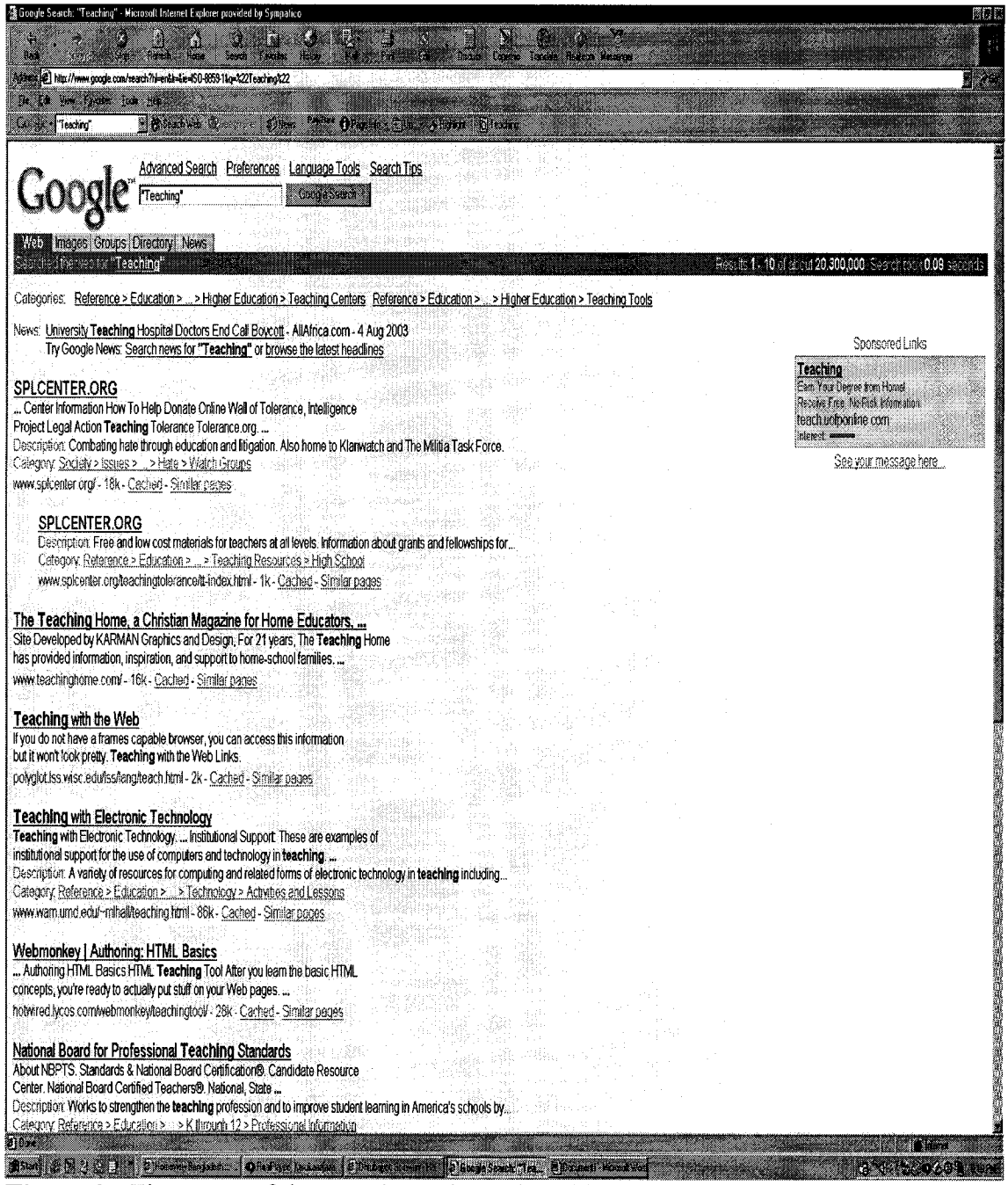

Figure 3: First page of the search result produced by Google for search topic "Teaching"

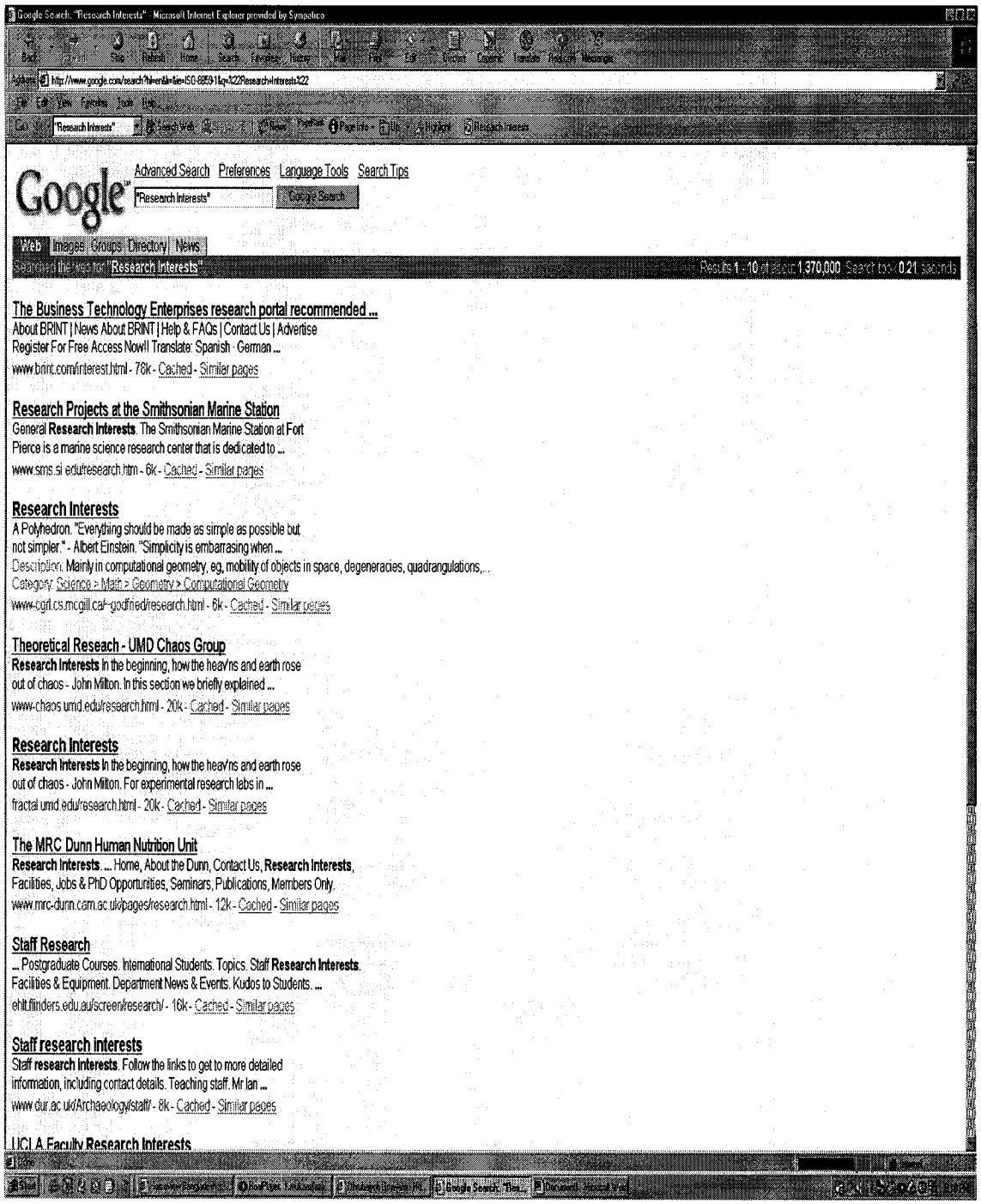

Figure 4: First page of the search result produced by Google for search topic "Research interests"

Now, let us search the documents that contain all of the above terms using the complex search string: **"Publications" AND "Teachings" AND "Research interests" AND "Courses"** in the advanced search of Google (see figure 5) taking the significant terms from the domain.

The findings are:

1. Among the top-ranked 10 documents in the search result, 6 documents are home pages of university faculties containing information about their publications, teachings, course offerings and research interests.
2. Also among the top-ranked 20 documents, 13 documents have information about university faculties containing information about their publications, teachings, course offerings and research interests.

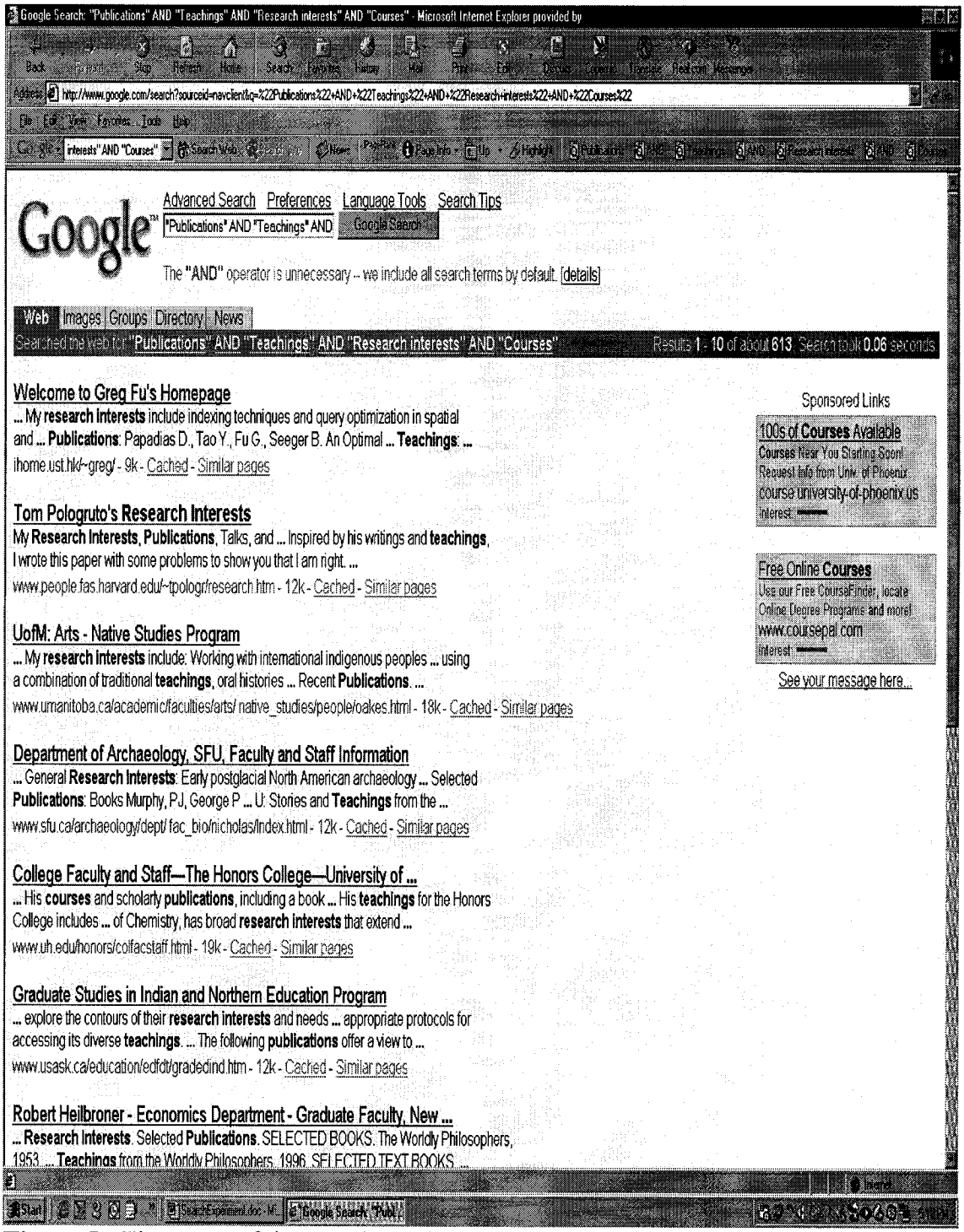

Figure 5: First page of the search result produced by Google for search topic "Publications" AND "Teachings" AND "Research interests" AND "Courses"

Therefore we can easily reach the conclusion that if any user wants to search the web for documents containing a topic term within a specific domain, the search phrase should

contain multiple terms that are significantly related to that domain including the search topic term. Again, matching domain significant terms in any document to find out the relevance of the document to the domain is suggested as the domain detection process in the motivation part of this major report. Therefore domain detection should be a part of the web search process that will effectively search a topic falling under any given domain. This will also eliminate the effort of the user to think about significant terms of any domain to construct an advanced search string to find accurate result.

4. Problem specification

In this major report, algorithms are formulated and discussed to solve and implement the following systems using ontologies:

1. A domain detector and annotator for HTML web documents.
2. An improved web document search engine.

The first module detects if any web document falls under a given domain. In doing so, it ranks the document to indicate how closely the document relates to the domain.

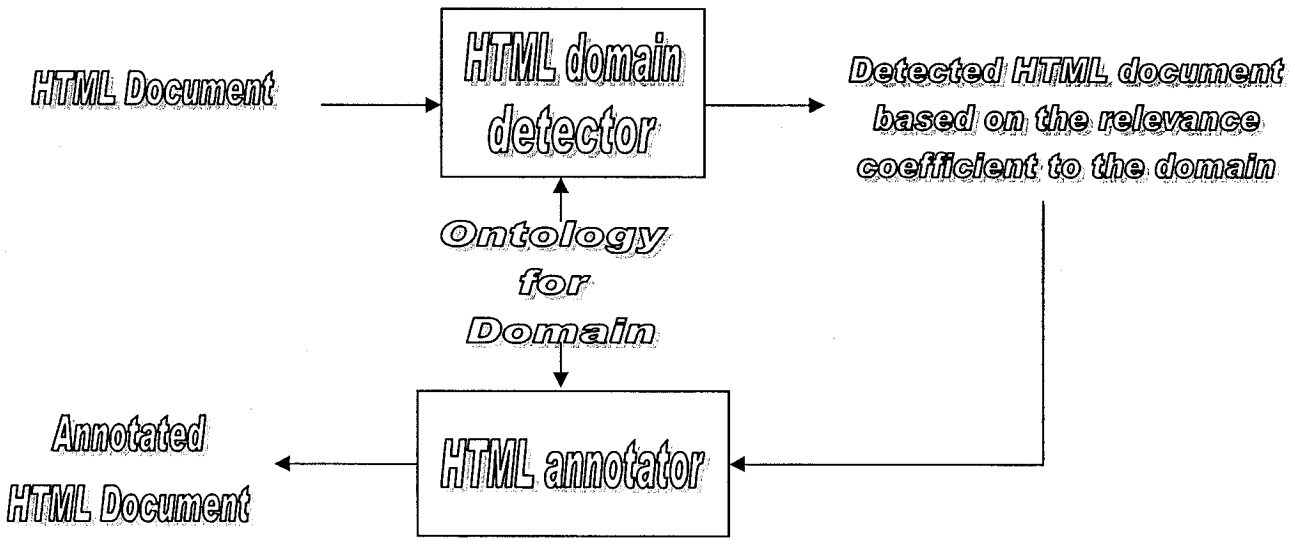

Figure 6: Block diagram for domain detector and annotator

The second module is an improved algorithm for searching the web that takes the search domain as input in addition to the search terms. It will search only the documents that contain the search terms and also belong to the given search domain. Taking the search domain into consideration improves the search result most of the time. The improved search engine module delegates the responsibility of domain detection to the domain detection module. It improves the reusability and modularity of the system code.

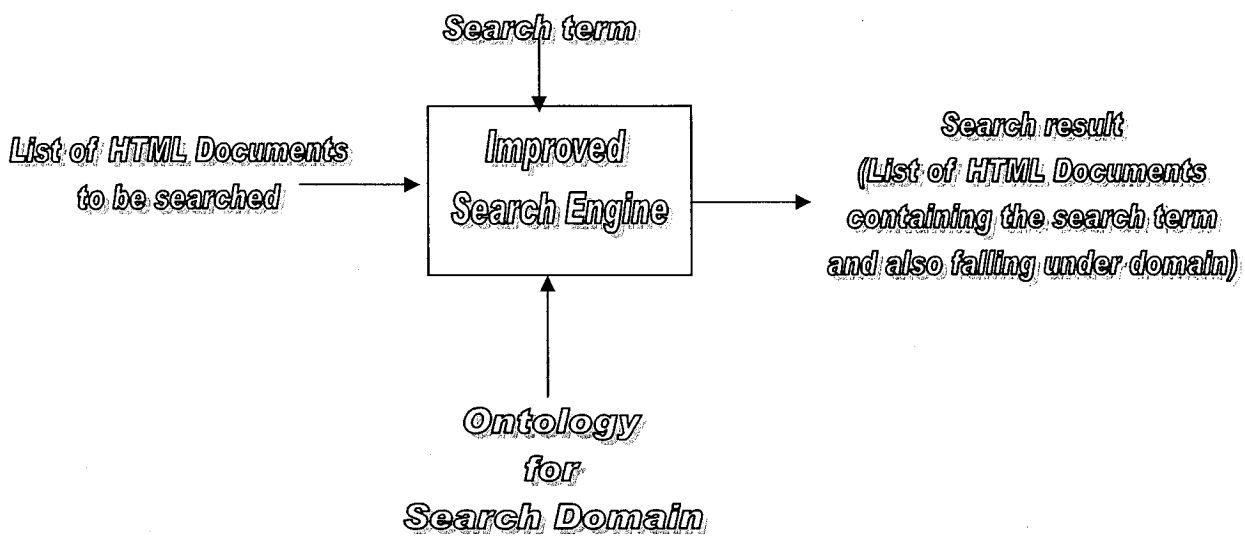

Figure 7: Block diagram for improved search engine

4.1 Test ontology selection

To test the implementation of the above modules, "university faculty home page" has been selected as the test domain. The corresponding ontology was developed in OilEd [6] and tested with the Racer [2] server. It was used for the following purposes:

- To find out the effectiveness and accuracy of the first module in deciding whether any given HTML document represents the home page of a University faculty member or not.
- To find out the effectiveness and accuracy of the second module in searching web documents that are only faculty home pages containing the given search term.

The system has been designed in a very generalized way such that it can be used with any ontology. Therefore we can assume, if the system works for a particular test domain, it will work for any other domain given that the corresponding ontology representing that domain is already designed, developed and deployed to be used by the system.

5. Problem solving approach

5.1 Domain detection and annotation module

Domain detection and annotation module has the following two steps:

- Initial setup
- Document detection and annotation

5.1.1 Initial setup

Pre-requisite: The ontology for the domain should already be designed, developed and loaded into an ontology server such as Racer server.

The initial setup method submits queries to the Racer server to get all the ontology terms from it. Subsequently more information for each ontology term like weight, definition, synonyms, all possible uses and variations, are added and stored in a lookup/mapping table in text file/XML file/database. The initial setup method also has the capability to read the stored mapping table into java objects and to serialize it to a binary file so that the object can be loaded by any java application to get the mapping table information. This object can be referred to as ontology object.

The initial step needs to be executed once when the ontology becomes available. We do not need to execute it again unless the ontology is enhanced. In the case when the ontology has been updated or enhanced, we execute the method to update the mapping table and subsequently re-generate the serialized ontology object.

5.1.2 Domain detection

The domain detection module has the following inputs:

- A list of documents to analyze and detect.
- The ontology object containing the term mapping information for the ontology that represents the domain.

The output is:

- The list of ranked documents according to the domain in descending order of rank so that the most relevant documents come first.

The domain detection process calculates the rank of each document to find out its relevance to the domain. First it scans the text content of a document and using exact string match it tries to find the list of ontology terms present in the document. If it fails to find any ontology term, it tries to find if any of its synonyms or different formations has been occurred in the document. In this way, the stemming and lexical semantics for the ontology terms are taken into consideration. We assume that if the term or any of its synonyms or variations occurs in the document, the ontology term is said to be present in the document and its weight is considered in document rank calculation. We can make a simple assumption that the relevance of any document to any particular domain is proportional to the number of different domain ontology terms present in the document. In other words, the higher the number of different ontology terms present in the document, the more it is close to the ontology domain. The process of determining the ontology term weights is explained in the section "Importance of term weight". As stated above, the ontology mapping table prepared in the initial step stored all these information like synonyms, different variations and weights for the ontology terms.

5.1.2.1 Calculation of rank

The domain detection is done by calculating the rank or relevance coefficient. Relevance coefficient is a real number calculated by adding the weights of the ontology terms that are occurring in the web document at least once. The formula is a simple weighted average as follows:

Relevance coefficient $= \Sigma W_i * F_i \ (i = 1..n)$

Where, $W_i =$ Weight of the the i-th term.

$F_i = 1$ if the i-th term is present in the web document at least once

0 if the i-th term is not present in the web document

There are n numbers of ontology terms.

Weight indicates how important the term is to determine the domain. The weight is a fraction ranging from 0.0 to 1.0. 0.0 weight means that the term is not significant at all in domain detection and so it is automatically cancelled out by the formula. The higher the value of the weight, the more is the significance of the term in determining the domain. So the more significant terms will be found in the document, the higher will be the rank according to the formula.

The term frequency is not considered in the formula because the total number of occurrences for any particular ontology term does not carry much information in determining the domain rather the presence or absence of any particular term carries more information. For example, any document unrelated to a particular domain can contain a significant domain term multiple times but in a related document it can occur only once. The term "publication" is a significant term for the domain "university faculty home page" and it has a higher weight. This term may occur in any university publication

web site more than once although it is not a university faculty web site. On the other hand, university faculty home pages usually contain the term "publication" once as a section header. If we incorporate the term frequency in the rank calculation, the rank for a university publication site may become more than a university faculty home page since the large weight of the term "publication" will be further multiplied. It is important to see how many significant terms are present in any document to determine its domain. The additional information about how many times each of the terms is encountered need not be incorporated in the rank calculation formula since it might make the calculated rank inaccurate.

5.1.2.2 Importance of term weight

The accuracy of the rank depends on the term weights. If we increase the weight of some significant terms then the ranking calculation is more accurate since the occurrence of more significant terms in a document increases the probability of the document falling under the domain. Also one term may be more significant than another in determining the domain. The weights of the terms can be determined with the following factors:

1. The knowledge about the significance of the terms.
2. Running a number of simulation runs with different weight distribution with varied sets of web documents and then examining the ranked list carefully to find out the optimum values for the weights that can produce the best result.
3. Supervised learning.

5.1.2.3 Cut-off rank

Also by setting a cut-off rank and removing the documents with ranks less than the cut-off value results in a ranked list of documents that exactly falls under the domain or very close to the domain eliminating irrelevant documents. This can also be determined with a number of simulation runs with different sets of web documents and then examining the ranked list carefully to find out the optimum value.

5.1.3 Document annotation

The domain detection module can be easily extended to annotate the web document with the corresponding ontology terms. Web document annotation can play a significant role as the first step from traditional syntactic web that we see presently as static HTML documents that is not very useful towards the semantic web. An annotated web document can be processed to derive more information about its different terms or concepts by search engines and software agents as opposed to web document that is not annotated.

It would be very hard and time consuming to annotate the vast collection of web sites available today manually. Availability of such a tool would solve the problem of annotation to a great extend. As more and more ontologies will be designed in the future, more and more web documents that fall under the domains represented by those ontologies can be automatically annotated if such a tool is available.

The domain annotation module has following parameters as inputs:

- A list of documents to annotate.

- The ontology object containing the term mapping information for the ontology that represents the domain.

The output is:

- The list of annotated documents according to the domain.

5.1.3.1 Annotation method

Some of the information that is stored in the mapping table is the definition of the ontology term. It is usually the URI where the definition and/or more information about the term can be found. During annotation, the text content of the document is scanned and all the occurrences of ontology terms along with their all possible uses (also stored in the mapping table) are converted to the HTML link corresponding to the URI.

5.2 Web search module

The web search module has the following two steps:

- Initial setup
- Document search by domain detection

5.2.1 Initial setup

Pre-requisite: Ontologies should be designed and available for all the members of the set of domains the user may be interested to search.

The initial setup is the same as the document detection/annotation module except it has to operate on multiple ontologies to generate the mapping tables in multiple ontology objects. The web search module will load the appropriate mapping table according to the user domain selection.

For simplicity, we assume that the only domain and ontology available is our test-case domain, which is university faculty home pages. The implemented system was tested assuming the user is interested to search on the test domain.

5.2.2 Document search by domain detection

The steps involved are:

- The user initiates a search with a search term and the domain of interest.
- The web module searches and finds a list of documents containing the search term.
- The web module passes the search result list along with the domain of interest to the domain detection module that filters out the documents not in the domain and then ranks the documents according to how close are they to the domain and returns the ranked list.
- The ranked list of documents is presented to the user in descending order of rank.

The module will also have the function to generate the advanced search string using all significant ontology terms and their possible uses from the mapping table. Any search engine or user can use this search string to produce better search results.

6. Ontology Design

One major part of the report was to design and implement a test ontology for testing the implemented system. Our test case was the ontology representing the information of home pages of university faculty.

6.1 Design approach

There is no "correct" methodology for developing ontologies. The best solution depends on the application and its future extension.

Some design approaches that I used in the ontology design are as follows:

1. Iterative design: I started with an initial design for the ontology and then revised and refined the evolving ontology and filled in the details. In each iteration, I checked the design solution with the goal that I wanted to achieve and refined the design accordingly.
2. Bottom-up approach: First I explored all the terms in university faculty home pages that are good candidates for ontology classes. Then I tried to define corresponding super-classes to derive the class hierarchies.

6.2 Design steps

Step 1: Determining the scope of the ontology

It is important to define the scope for the ontology before the design. The scope will be determined according to the goal of the ontology meaning how the ontology will be used.

For example, here are some important and informative features of university faculty home page, included in the ontology design:

- Course
- Research interest
- Publication

On the other hand, here are some unnecessary features of university faculty home page, left out in the ontology design since those will not be used or looked at frequently during any information search:

- Previous industry work experience
- Hobbies
- Photo Gallery

Analyzing in this way, only important features of university faculty home page were included in the ontology design process so that it can be useful to provide necessary information that is demanded frequently.

One way to determine the scope of the ontology is to sketch a list of competency questions and test if the ontology is able to answer these questions.

Some competency questions for the designed ontology are as follows:

1. What are the courses being offered by any university faculty member in the coming fall semester?
2. Get the list of all faculties who have more than 10 publications this year.

3. What are the research projects being supervised by any faculty?
4. Get all the publications for any faculty for the current year.
5. Get all the list of faculty members having research interests in a particular domain.

The ontology designer should derive a list of such questions to define the scope of the ontology so that the final design should be able to answer all the questions. For the university faculty homepage domain, a list of such questions was derived to define the scope and the final ontology was tested to make sure the information captured in the design is sufficient enough to answer those questions.

So, it is important that the design should be comprehensive enough to include all the necessary information that can be asked by any ontology user and at the same time should not include any unnecessary information.

Step 2: Enumeration of all available concepts and slots to derive the Taxonomy

At the second step, all the important terms or concepts were collected by visiting many university faculty home pages of different universities around the world in different subject areas. Then the properties or characteristics for each of these concepts were derived. Through the iterative design approach both the concepts and properties were revised and refined according to the scope as explained in the first step. The concepts are mapped to classes and the properties are mapped to slots in the ontology design.

Step 3. Derivation of the class hierarchies

There are several possible approaches in developing a class hierarchy:

- Top-down development process
- Bottom-up development process
- Combination of top-down and bottom-up approaches

The **bottom-up** approach was used in defining class hierarchies. First all of the more specialized concepts in the domain were collected to define the bottom-most classes of the class hierarchy and then more general classes were formulated to define the complete class hierarchies. In the top-down approach, more generalized classes are defined first and then the more specialized classes are derived as the children of the generalized classes.

Step 4. Derivation of class relationships

Next step was to define the relationships among the derived classes. There are mainly two types of relationships:

1. Inheritance: **is_a** relationship (represented by the empty arrow in Figure 8)
2. Composition: **is_part_of** relationship (represented by the black filled diamond in Figure 8)

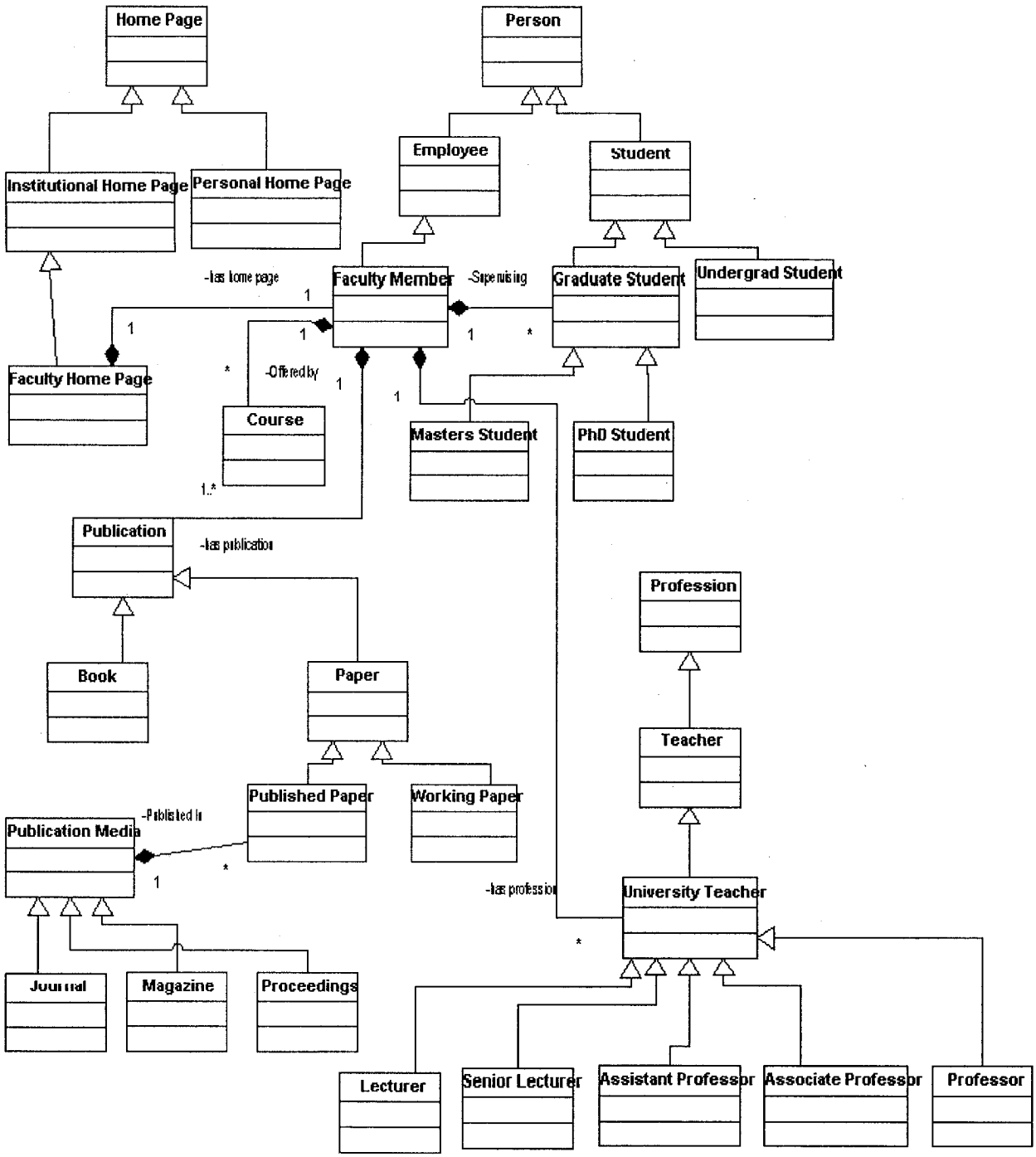

Figure 8: The class diagram showing major class hierarchies and the relationships between classes for University faculty home page ontology

Step 5. Implementation of the ontology using OilEd and testing it using Racer and Rice [7]

OilEd was used to construct the ontology which stores the ontology in DAML format. With the iterative approach the ontology was revised and refined a couple of times. After completing the ontology, the following tests were conducted.

Tests:

1. Using Racer, the ontology was verified to find out inconsistencies in the design and subsequently the inconsistencies were corrected using OilEd.
2. Using Rice, a couple of T-Box [2] queries were executed to verify all the necessary T-Box information were present and valid. No A-Box [2] queries were done since we did not defined any instance. The reason for this is that only the T-Boxes are enough to test the system descripbed in this report and we do not need any instance of the ontology.

7. Implementation

The java API for the implemented system is listed below; other systems, search engines and software agents can call the functions in the API or can extend it.

7.1 Java API for the implemented system

1. List detectResult(Ontology ontologyDomain, List documentsToDetect)

The function has 2 parameters:

a. ontologyDomain: The ontology object representing the domain the user is interested in.

b. documentsToDetect: The document list to be detected.

This function analyzes the list of documents passed as a parameter and returns a sorted list of documents in descending order of rank indicating the relevance of the documents to the given domain.

2. List searchResult(String topic, Ontology ontologyDomain, List documentsToSearch)

The function has 3 parameters:

b. topic: The topic the user wants to search.

c. ontologyDomain: The ontology object representing the domain the user is interested in.

d. documentsToSearch: The document list to be searched.

This function searches the list of documents passed as a parameter and returns a sorted list of documents in descending order of rank according to the relevance to the given

domain. Each of the documents in the returned list will contain one or more of the given search topic.

7.2 Selection of programming language

The Java programming language has been used to implement the system. The reason for using Java is to take the advantage of the object oriented analysis, design and availability of a large number of Java APIs that include very useful classes and utilities. For example, the collection API includes classes implementing very useful data structures such as Linked List, Vector, Tree, Hashtable etc. The classes StringTokenizer and StringBuffer are very useful in parsing that is heavily used in the system. Also useful APIs for file processing, XML document parsing and lots of other utilities are available from Sun, the creator of Java and other third parties.

All these facilities will minimize the system design and implementation time. Also it will make code simple, precise, well designed and organized with increasing readability, maintainability and scalability.

7.3 Implementation details

7.3.1 Initial configuration

For both the domain detection and the search module, there exists an initial configuration process to generate the java ontology object that represents the mapping table for the ontology terms.

The steps are:

1. The initial query module submits the query to the racer server to get all the ontology terms and lists all the terms into an initial raw ontology text file. The sub-steps are:
    (a) The ontology to be processed should be loaded into the ontology editor OilEd. For our experiment, the university faculty home page ontology was loaded into OilEd.
    (b) OilEd should be connected to the reasoner, the Racer server.
    (c) Now the ontology should be submitted to the Racer server for classification.
    (d) Once the Racer server has received the ontology, then the java initial query module can submit any query to get information from the ontology through the Java Racer API.
    (e) The query module then iterates through the result set it receives from the Racer server and writes all the ontology terms each in a separate line of an initial raw text file.
2. The initial raw text file should now be processed by a user or process to add more information about the terms to the text file according to agreed upon format/protocol. The additional information is as follows:
    (a) URI link to the definition and information about the term (to be used for the annotation process).
    (b) Weight to indicate the significance of the term in determining the domain (to be used by the domain detection process).

(c) All possible uses and forms of the term in HTML documents (to be used by the domain detection process).

3. The loader module then reads the information from the updated text file, creates the ontology java object and then serializes the object into a binary file.

Therefore, the system has been designed as a series of transformations so that any third party software or process can access and use the outcomes from each of the transformations (see Figure 9).

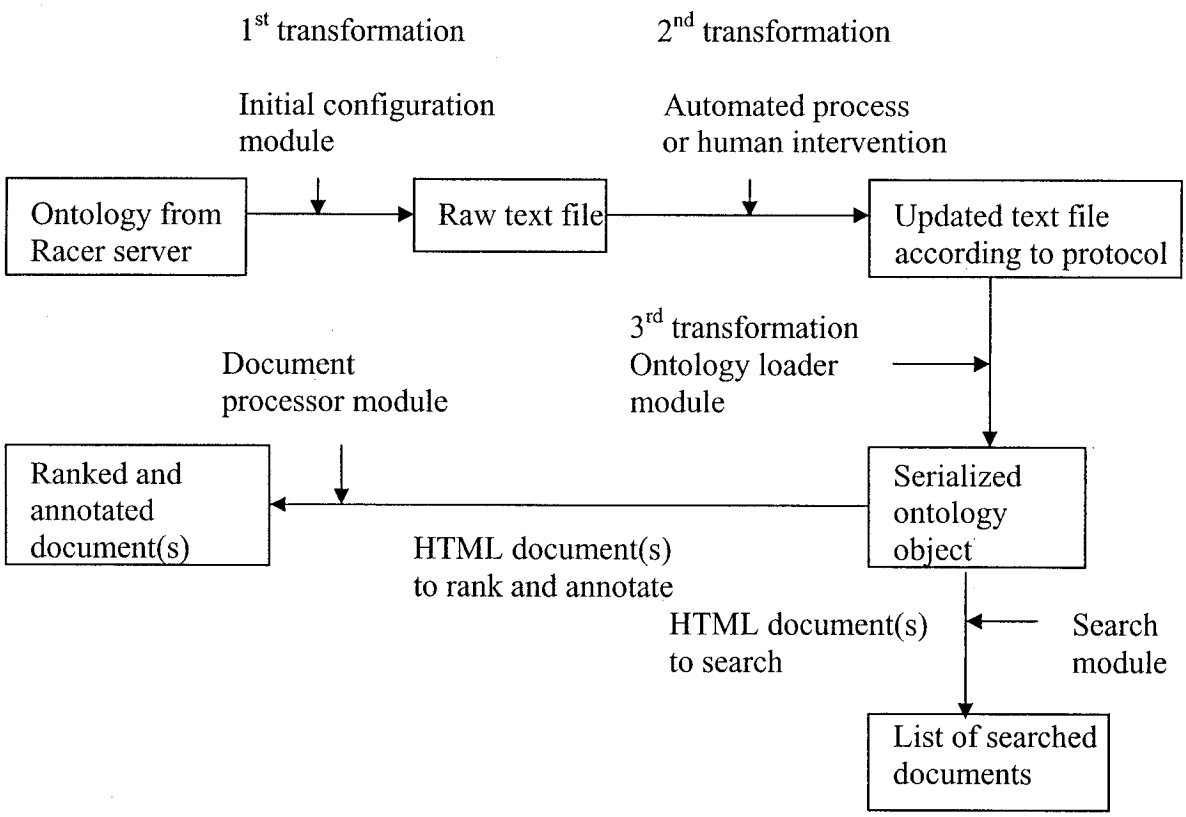

Figure 9: The transformation steps for the implemented system

Some advantages for the transformation steps are:

1. If the ontology is updated or enhanced, the ontology text file can be updated. Subsequently the serialized ontology object is re-created by the executing the loader module. So, when the document processor module executes next time it

use the updated serialized ontology object and therefore the updated ontology will be used for ranking and annotating documents.

2. Also the updated text file or the serialized ontology can be transported over the network to be used by any other processes or agents in a distributed computing environment.

The ontology text file plays the central role for the system and its format is discussed below:

The ontology text file contains a list of text lines or records each representing one ontology term extracted from the ontology uploaded in the racer server. So if there are n terms in the ontology, there will be n number of lines each carrying some important pieces of information about the corresponding term. Each line is a semicolon-separated list of tokens as follows:

<ontology term>;<url>;<weight>;<comma separated list of all possible term uses >

Where,

- 1st token is the ontology term.
- 2nd token is the URL to be used to find the definition of the term and so can be used to annotate the term in any web document.
- 3rd token is the weight which is a ratio ranging from 0.0 to 1.0 for the ontology term indicating how significant is the term for the domain it represents and therefore can be used for rank calculation. If the weight is 0.0, then the system will ignore the ontology term meaning it is not significant at all in determining the domain.

- 4th token is a comma-separated list of tokens where each token represents a particular use of the term.

For example, the following line represents the information about the ontology term "course" extracted from the faculty home page ontology:

course;http://www.faculty-ontology.com/course;1.0;courses,offering,current courses

It means that,

- The name of the ontology term is "course".
- Its definition can be found in the link http://www.faculty-ontology.com/course.
- Its weight is 1.0.
- Its synonyms and other formations or uses are "courses", "offering", "current courses" which can occur in any university faculty home page instead of the term "course".

Similarly the line representing the term "research" can be as follows:

research;http://www.faculty-ontology.com/research;1.0;current research,interests

So, the application follows the protocol described above to read the information about ontology terms from a text file. Since XML is a W3C recommended standard for information exchange [5], the application should use XML to store and retrieve the ontology term information. But due to time constraint, this feature could not be added to the application.

7.3.2 Web document detection and annotation

The system implementing document domain detection and annotation is divided into the following steps:

1. The ontology object representing the domain to be detected is loaded from the binary file into the Java object or mapping table containing information for the ontology terms for that domain.
2. Each of the HTML documents is scanned to derive the frequency table for the occurrences of the ontology terms.
3. The ranking formula is then applied to derive a rank for the document. The ranking formula is the sum of the weighted frequencies using the frequencies and weights of the ontology terms occurred in the document.
4. The annotation of the document is simply the process of converting the occurrences of the ontology terms in the document with the corresponding URL link where more information is available for any term.

7.3.3 Web document search

The system implementing the document search is divided into the following steps:

1. The input for this module is the user search term and the selected domain. The Ontology object representing the selected domain is loaded from the object binary file into the Java object or mapping table containing information for the ontology terms for that domain.
2. The given list of documents is scanned one by one to produce the output list of documents that contain the search term.
3. Now the output list of documents and the selected ontology object that represents the domain are passed to the document detection module to rank the documents according to the degree of similarity to the selected domain.

Therefore the output list of the ranked documents from the document detection module represents the final list of searched documents containing the search term and also falling under the selected domain.

8. Result

The implemented document domain detector and annotation module was tested with 12 sample web documents to find out its accuracy. The sample test domain was faculty home page domain and sample documents were selected from the following categories:

1. Fully Related (Web document that is a faculty home page).
2. Closely Related (Web document that has similar information and partially relates to Faculty home page).
3. Unrelated (Web document that may have similar terms or information but not related to faculty home page at all).

The sample web documents were:

Document URL	**Category (with respect to domain)**
http://www.cs.concordia.ca/~haarslev/	Fully Related
http://www.eecs.berkeley.edu/~bartlett/	Fully Related
http://www.columbia.edu/cu/newsstand/	Closely related
http://www.man.ac.uk/about/public.html	Closely related
http://www.asbpe.org/	Unrelated
http://hci.stanford.edu/winograd/	Fully Related
http://www.vu.msu.edu/site/courses.php	Closely related
http://www.harvard.edu/academics/research.html	Closely related
http://pantheon.yale.edu/~ym49/	Fully Related
http://www.research.ibm.com/	Un related
http://www.docnmail.com/	Unrelated
http://www.sispain.org/english/course/	Unrelated

The significant terms with respect to the domain are allocated more weights whereas the insignificant terms are allocated less weights. For example, some of the significant terms for university faculty home pages are "Research", "Publication" etc. and so 5.0 was allocated. Insignificant terms are allocated 1.0 or less that 1.0.

The ranked result was as follows in descending order of rank:

Document URL	Category	Rank
http://pantheon.yale.edu/~ym49/	Related	34.5
http://www.cs.concordia.ca/~haarslev/	Related	21.1
http://www.columbia.edu/cu/newsstand/	Closely Related	18.8
http://hci.stanford.edu/winograd/	Related	17.5
http://www.man.ac.uk/about/public.html	Closely Related	16.9
http://www.eecs.berkeley.edu/~bartlett/	Related	15.1
http://www.vu.msu.edu/site/courses.php	Closely Related	13.1
http://www.harvard.edu/academics/research.html	Closely Related	12.7
http://www.research.ibm.com/	Un related	12.1
http://www.asbpe.org/	Unrelated	5.0
http://www.sispain.org/english/course/	Unrelated	3.0
http://www.docnmail.com/	Unrelated	2.2

In the result, the 2 highest ranked scores go to documents those are fully related. Among the high ranking 8 documents, all are fully or closely related to the domain and the

unrelated documents receive the lower ranked scores. This demonstrates the accuracy of the implemented system for domain detection on this experiment.

Also the annotated web documents are generated correctly with correct URL links embedded in the HTML for the encountered ontology terms.

The same list of web documents is provided to the web document search module with input search term "Research" while the only available ontology is the test case domain "University faculty home page" . The searched result was as follows:

Document URL	**Category**	**Rank**
http://pantheon.yale.edu/~ym49/	Fully Related	34.5
http://www.cs.concordia.ca/~haarslev/	Fully Related	21.1
http://www.columbia.edu/cu/newsstand/	Closely Related	18.8
http://hci.stanford.edu/winograd/	Fully Related	17.5
http://www.man.ac.uk/about/public.html	Closely Related	16.9
http://www.vu.msu.edu/site/courses.php	Closely Related	13.1
http://www.harvard.edu/academics/research.html	Closely Related	12.7
http://www.research.ibm.com/	Un related	12.1

Analyzing the result, we see that all 8 documents in the searched result contain the search term "Research" and 7 documents are either closely or fully related to the test domain and 1 document is not related to the domain. The result was displayed in descending order of rank so that documents fully or closely related to the domain come first.

The content of the ontology term text file that has been used to provide the ontology term information to the application is given below:

Research Group;http://www.faculty-ontology.com;5.0;research,current research,research interest
Conference;http://www.faculty-ontology.com;3.0;Upcoming Conference,Upcoming Conferences
Publication Media;http://www.faculty-ontology.com;1.0;
Masters Thesis;http://www.faculty-ontology.com;4.0;
Journal;http://www.faculty-ontology.com;1.0;
Faculty Member;http://www.faculty-ontology.com;5.0;
Doctoral Thesis;http://www.faculty-ontology.com;4.0;
Teacher;http://www.faculty-ontology.com;5.0;faculty
1st Quarter;http://www.faculty-ontology.com;1.0;
Academic Award;http://www.faculty-ontology.com;1.0;Award,Awards,Academic Awards,list of awards
Publication;http://www.faculty-ontology.com;5.0;Publications,current publications,list of publications
University Teacher;http://www.faculty-ontology.com;2.0;
School;http://www.faculty-ontology.com;1.0;
Working Technical Report;http://www.faculty-ontology.com;1.0;
Academic Position;http://www.faculty-ontology.com;3.0;
Seminer;http://www.faculty-ontology.com;3.0;Upcoming seminers,seminers
Associate Professor;http://www.faculty-ontology.com;3.0;
3rd Quarter;http://www.faculty-ontology.com;1.0;
Presentation;http://www.faculty-ontology.com;1.0;
University;http://www.faculty-ontology.com;1.0;
Masters Student;http://www.faculty-ontology.com;5.0;supervising masters students
Institute;http://www.faculty-ontology.com;1.0;
Summer;http://www.faculty-ontology.com;1.0;
Professor;http://www.faculty-ontology.com;3.0;
Computer Network;http://www.faculty-ontology.com;0.2;
Article;http://www.faculty-ontology.com;1.0;Articles,published articles
2nd Quarter;http://www.faculty-ontology.com;1.0;
4th Quarter;http://www.faculty-ontology.com;1.0;
Published Paper;http://www.faculty-ontology.com;5.0;Published Papers,papers,paper
Academic Event;http://www.faculty-ontology.com;0.2;
Magazine;http://www.faculty-ontology.com;0.1;
Computer Science;http://www.faculty-ontology.com;0.5;
Talk;http://www.faculty-ontology.com;2.0;talks,upcoming talk,upcoming talks
Thesis;http://www.faculty-ontology.com;1.0;

Economics;http://www.faculty-ontology.com;0.2;

Academic Period;http://www.faculty-ontology.com;2.0;

Proceedings;http://www.faculty-ontology.com;3.0;

Assistant Professor;http://www.faculty-ontology.com;3.0;

Course;http://www.faculty-ontology.com;2.0;courses,courses offered,current courses

Program;http://www.faculty-ontology.com;0.2;

Profession;http://www.faculty-ontology.com;0.1;

PhD Student;http://www.faculty-ontology.com;5.0;PhD students,supervising PhD students

Undergrad Student;http://www.faculty-ontology.com;4.0;Undergrad Students

NLP;http://www.faculty-ontology.com;0.2;Natural Language Processing

AI;http://www.faculty-ontology.com;0.2;Artificial intelligence

Working Paper;http://www.faculty-ontology.com;5.0;

HomePage;http://www.faculty-ontology.com;0.1;web site,home page,Personal Home Page

Lecturer;http://www.faculty-ontology.com;3.0;

Fall;http://www.faculty-ontology.com;1.0;

Subject Area;http://www.faculty-ontology.com;0.5;Field

Winter;http://www.faculty-ontology.com;1.0;

University Teacher Home Page;http://www.faculty-ontology.com;8.0;

Quarter;http://www.faculty-ontology.com;0.2;

Graduate Student;http://www.faculty-ontology.com;5.0;graduate students,grad students

Position;http://www.faculty-ontology.com;0.5;designation

Academic Department;http://www.faculty-ontology.com;0.5;department

Senior Lecturer;http://www.faculty-ontology.com;3.0;

Semester;http://www.faculty-ontology.com;0.5;

9. Limitation of the implemented system

The algorithm proposed in the major report calculates rank score using the matching ontology terms. So document rank score depends on the total number of different matching terms in the document content. This is true for the web documents falling under the domain represented by the ontology. This may also be true for the web documents not falling under the domain but have a high number of matching ontology terms in their content making their rank scores high.

10. Related work

1. Word-Net

In Word-Net [3], when any user searches for a word, a list of senses is presented to the user so that the appropriate sense is selected. Then the meaning for the searched word for the selected sense is displayed. Word-Net is a large ontology to identify any potential ambiguities, and asks the user to identify the correct concept. For example, if the user enters the word "tank", Word-Net asks the user to identify the correct concept (e.g. a storage tank, or an army tank). Then the search is re-formulated by automatically appending the negation of terms closely related to the wrong concept. For example, if the desired concept is storage tank, then words strongly associated with 'army' are excluded in the search result.

2. Yahoo!

Yahoo! uses an ontology as an index into some kind of repository. All items in the repository are linked to items from the index. An ontology author creates an ontology (typically a simple taxonomy with relations between terms). A knowledge worker then uses this taxonomy to help limit the scope of their search. Yahoo! uses these terms to locate relevant documents in a repository.

11. Future extension

The application of ontologies in the area of domain detection, annotation and search for web documents definitely demands more research.

One of the most necessary future extensions of the work represented by this report is to continue to do research on how to incorporate more information about the ontology classes/terms within the ontology. This will help the automation of the mapping table generation and to reduce user intervention in extending the mapping table. As described in this report, the mapping table is generated by capturing all the ontology terms and then by adding more information for each term such as weight, uses, definition etc. that requires human intervention. If this additional information about the ontology terms can be integrated into the ontology structure, we could generate the final mapping table without any human intervention.

One of the solutions to this is to find a mechanism to put the additional information about the classes/terms within the structure of the ontology itself. The mechanism may be to define special classes and/or axioms. Another way may be to design a second meta-ontology that will contain the meta-data such as additional information about the ontology terms.

Another extension may be to develop a NLP system that can capture all possible uses and variations of the ontology terms by scanning corpora (the standard texts used for NLP) and update the mapping table that has been proposed by this major report for domain

detection. This will eliminate the manual intervention to find out and update the same information.

The capability to read XML file to retrieve information about ontology terms can be added as a new feature to the implemented system. Since XML is a W3C recommended standard for information exchange, using XML would facilitate communicating with various other systems to get the ontology term information.

In rank calculation that is discussed in the major report, we can consider other factors such as length of the document. For a large document, the probability of encountering ontology terms is more likely than in a small document. In a small document we may not encounter significant ontology terms although the document may fall under the ontology domain. As a future extension, experiments can be done to find out how the document length effects the rank calculation to incorporate the factor in calculation.

12. References

1. Natalya F. Noy and Deborah L. McGuinness: Ontology Development 101: A Guide to Creating Your First Ontology, Stanford University, Stanford, CA, 94305, noy@smi.stanford.edu and dlm@ksl.stanford.edu (URL: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html, last visited: April 12, 2004)

2. *Volker Haarslev, Ralf Møller*: Description of the RACER System and its Applications In Proceedings of International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August 2001.

3. Word-Net, URL: http://www.cogsci.princeton.edu/cgi-bin/webwn, last visited: April 12, 2004

4. W3C OWL reference, URL: http://www.w3.org/News/2004#item14, last visited: April 12, 2004

5. W3C XML reference, URL: http://www.w3.org/XML/, last visited: April 12, 2004

6. OilEd, URL: http://oiled.man.ac.uk/, last visited: April 12, 2004

7. Rice, URL: http://www.b1g-systems.com/ronald/rice/, last visited: April 12, 2004

8. Apache logging API, URL: http://logging.apache.org/log4j/docs/documentation.html, last visited: April 12, 2004

13. Appendices

13.1 Application configuration file

The file **resources.properties** is consulted by the application to get the configuration values and input parameters. The keys in the configuration file are explained below:

Key	Sample value	Description
working.directory	c:\\ontology\\	Working or base directory. Assumes current directory as the working directory if no value is specified for this key.
Output.directory	c:\\ontology\\	Output directory to write annotated files. Assumes current directory as the working directory if no value is specified for this key.
ontology.term.txt.file.name	raw_ontology.txt	Initial ontology text file containing the list of all the ontology terms generated by getOntologyFromRacerServer.bat
ontology.txt.file.name	ontology.txt	Updated ontology text file containing all the ontology terms with additional information for each term.
serialized.ontology.object.file.name	ontology.obj	Serialized ontology object file name.
input.file.list	a.html, b.html, c.html	List of comma separated input web documents for which we need to do one or more of the following: 1. Detect domain 2. Annotation 3. Search any given term

generated.file.name(s).after.annotation	annotated_a.html, annotated_b.html, annotated_c.html	List of comma separated output file names for the annotated web documents
url.of.input.file.list	www.domain.com/a.html, www.domain.com/b.html, www.domain.com/c.html	List of comma separated URLs that corresponds to the input web document list given by input.file.list
description.of.input.file.list	document a, document b, doicument c	List of comma separated descriptions that corresponds to the input web document list given by input.file.list
Search.term	Research	The term to search in the list of web documents given by input.file.list
racer.server.IP	127.0.0.1	Racer server IP address.
racer.server.port	8088	Racer server port.

13.2 Application Deployment

13.2.1 Deployment steps

1. All the necessary files required for the application installation are zipped into the file **ontology.zip**. This needs to be un-zipped into a suitable directory to get the following list of files:
    - resources.properties (Application configuration file)
    - ontology.jar (Application JAR file)
    - required libraries (Required to run the application)
2. Edit the property file resources.properties to set up appropriate values for the keys. The keys are explained in the "Application configuration file" section above.

3. **log4j.log** file is the log file produced by log4j logging library. log4j.log file should be edited if the logging configuration needs to be changed. More details information can be found in the log4j documentation link of apache web site. [8]

13.2.2 Application execution instructions

13.2.2.1 Ontology mapping table preparation

1. Launch OilEd and load the target ontology.
2. Start the racer server, connect it from OilEd and classify the ontology in OilEd by submitting it to the racer server.
3. Run the batch file getOntologyFromRacerServer.bat to query racer server to get the ontology taxonomy and then to generate an initial mapping table listing all the ontology terms in a text file.
4. Update the initial mapping text file according to the protocol described in the major report to add additional information for the ontology terms.

13.2.2.2 Web document domain detection and annotation

1. Execute the steps in the "ontology mapping table preparation".
2. Configure the corresponding input parameter keys as explained in the "Application configuration file" section above.
3. Run the batch file rankAndAnnotateDocument.bat to rank and annotate one or more web documents according to the domain represented by the ontology submitted to the racer server in step 2 of "Ontology mapping table preparation". A

ranked list of detected document will be displayed and logged. The corresponding annotated documents will be generated in the output directory.

13.2.2.3 Web document search

1. Execute the steps in the Ontology mapping table preparation.
2. Configure the corresponding input parameter keys as explained in the "Application configuration file" section above.
3. Run the batch file searchWebDocument.bat to search the input term in the listed web documents and a ranked list of searched documents containing the term will be displayed and logged.