Nova Southeastern University
NSUWorks

CEC Theses and Dissertations | College of Engineering and Computing

2008

Use and Analysis of Expected Similarity of Semantic Web Ontological Annotations

Joe Lynn Look
Nova Southeastern University, look32080@gmail.com

This document is a product of extensive research conducted at the Nova Southeastern University College of Engineering and Computing. For more information on research and degree programs at the NSU College of Engineering and Computing, please click here.

Follow this and additional works at: http://nsuworks.nova.edu/gscis_etd

Part of the Computer Sciences Commons

Share Feedback About This Item

NSUWorks Citation

Joe Lynn Look. 2008. *Use and Analysis of Expected Similarity of Semantic Web Ontological Annotations.* Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, Graduate School of Computer and Information Sciences. (684) http://nsuworks.nova.edu/gscis_etd/684.

This Dissertation is brought to you by the College of Engineering and Computing at NSUWorks. It has been accepted for inclusion in CEC Theses and Dissertations by an authorized administrator of NSUWorks. For more information, please contact nsuworks@nova.edu.

Use and Analysis of Expected Similarity of Semantic Web Ontological Annotations

by

Joe Lynn Look

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Science

Graduate School of Computer and Information Sciences
Nova Southeastern University

June 2008

We hereby certify that this dissertation, submitted by Joe Lynn Look, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Junping Sun, Ph.D. Chairperson of Dissertation Committee	Date
James Cannady, Ph.D. Dissertation Committee Member	Date
Michael Laszlo, Ph.D. Dissertation Committee Member	Date

Approved:

Edward Lieblein, Ph.D. Dean	Date

Graduate School of Computer and Information Sciences
Nova Southeastern University

2008

An Abstract of the Dissertation Submitted to Nova Southeastern University in Partial

Fulfillment of the Requirements of the Degree of Doctor of Philosophy

Use and Analysis of Expected Similarity of Semantic Web Ontological Annotations

by
Joe Lynn Look

June 2008

This dissertation studied various means of calculating similarity in the annotations of web pages compared to the similarity of the document text. A software tool, named Semantic Web Analysis of Similarity (SWAS), was developed and utilized to perform the analysis of similarity in annotated documents published from the first three years of the International Semantic Web Conference. Rules concerning the ontological concepts of the documents were specified and these rules as well as other parameters were varied to determine the effect on overall similarity measures. Traditional measures of similarity as well as enhanced measures of similarity proposed for this study were evaluated. A proposal was made concerning use of similarity measures to evaluate the consistency of semantic annotation for documents published through a Semantic Web portal.

Acknowledgments

First and foremost, I would like to acknowledge the contribution of my committee chair, Dr. Junping Sun as well as the other members of the committee, Drs. James Cannady and Michael Laszlo. The knowledge gained from their courses, as well as those of other instructors in the doctoral program, prepared me to do this research. Without the advice, support and patience of these people, this dissertation would not have been possible.

I would also like to acknowledge the contributions of those Semantic Web researchers who have published their research in a spirit of scholarly learning. Special thanks go to Dr. York Sure for his suggestion of using the data from the International Semantic Web Conference, and to Dr. Prasanna Ganesan for his advice concerning application of the generalized cosine similarity measure.

On a personal note, I would like to acknowledge the contribution of my husband, David Look, whose encouragement and assistance with home responsibilities made it possible for this research to be accomplished. I would also like to thank those fellow students at Nova Southeastern University who provided encouragement throughout the doctoral program.

Table of Contents

List of Tables

Tables

List of Figures

Chapter 1

Introduction

Problem Statement

The Semantic Web is a redesign of the World Wide Web (WWW) based on standards proposed by the World Wide Web Consortium (W3C). Another web development trend is Web 2.0, which is intended to increase efforts at collaboration to deliver services designed to take advantage of the structure of WWW. A major focus of Web 2.0 is the development of applications that combine various types of data that are located in different places on the Web. This combination of data from data sources is usually known as "mash-ups" (O'Reilly, 2005). Many aspects of the Semantic Web also involve data sharing, so the two trends complement each other in many aspects, but the Semantic Web is based on a consistent model and tools for defining and using related data (W3C, 2008). There are many facets to the Semantic Web, but one of the key elements is the development of Semantic Web Documents (SWDs) created according to those standards that contain ontological annotation, which will allow machine agents to understand more about the meaning of documents.

The annotation needed for the Semantic Web is provided by inserting markup tags that provide an understanding of the relationships of key concepts, rather than simply the keyword tags in common use today. This semantic markup is intentional knowledge that the author of the Web page is explicitly providing through annotation based on a specific ontology to aid in searching for and extracting information from the Web. In addition to the annotated documents, another feature of the Semantic Web is the creation of enhanced web services being developed to utilize those annotations.

The term ontology has many meanings in various contexts. In the introductory section of the requirements document defining use cases and requirements for the Semantic Web, Heflin (2003) gives the following description of ontology as it is applied to the Semantic Web:

> An ontology defines the terms used to describe and represent an area of knowledge. Ontologies are used by people, databases, and applications that need to share domain information. Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and knowledge that spans domains. In this way, they make that knowledge reusable.

In addition to the components necessary to build a Web that is more navigable by machine agents, much research and development have been done in the area of tools for the Semantic Web. The W3C has standardized the basis for the Semantic Web to be the Resource Description Framework (RDF). This framework brings together various applications using Extensible Markup Language (XML) for syntax and universal resource identifiers (URIs) for naming concepts and properties. Considerable research has been done to determine the ontological features that should be present in a Semantic Web language. Various languages have been developed to create Semantic Web ontologies based on that research. The Web Ontology Language (OWL) is the latest one proposed by W3C to define ontologies and implement semantic annotation of SWDs. Currently there are tools that help knowledge engineers build Semantic Web ontologies and tools to help web authors use those ontologies to annotate documents. These tools do not assist in determining if those annotations are similar to other like documents already existing in the Semantic Web. In order to be truly valuable, semantic annotations should be consistent as well as syntactically correct. Some of the tools to evaluate the appropriateness of annotations included in SWDs do not currently exist. The purpose of this dissertation is to analyze measures of similarity for semantic annotation in hopes this research could be applied to develop tools to assist web authors in validating the appropriateness of the annotations. Valid similarity measures could be used as a basis for a tool that would compare new web documents to an existing set of appropriately marked-up documents to validate the proposed annotation.

Relevance and Significance

The popularity of search engines today is a testament to the importance of finding information on the Web. Research projects have been devoted to finding better ways to classify information so it can be found more readily. The Automatic Classifier for Internet Resource Discovery (ACIRD) system utilizes a semantic approach to mining term associations (Lin, Shih, Chen, Ho, Ko, & Huang, 1998). This system was developed by manually classifying targeted documents and using this information as training data for the information retrieval (IR) system that was developed to automatically classify a larger set of documents. Based on the classifications, term associations are inferred. Haustein and Pleumann (2002) show how the semantic inferences found using this system can lead to increased knowledge about Web documents. Meaningful annotations included in SWDs will serve to classify documents just as the ACIRD system did, but only if those documents are annotated carefully and validly. Another established fact is that for the Semantic Web to have true usefulness there must be a sufficient base of accessible annotated pages (Haustein & Pleumann, 2002).

Up until now, there has not been widespread use of semantic annotation in Web documents. One reason frequently mentioned for the lack of annotation is that many Web authors do not fully understand the knowledge management concepts that must be applied for the annotations to be useful. Another reason is that there is currently little direct benefit to the Web author to enhance the page semantically (Haustein & Pleumann, 2002). Unless tools are designed that will simply and accurately assist with annotation of Web pages and allow for verification of those annotations, the Semantic Web will not garner the widespread use needed to make it beneficial, either academically or commercially. For Semantic annotation to become more widely used there must be some obvious reward for Web authors who include it in their documents. Many Web authors today go to great lengths to design Web pages that will be recognized by search engines. Web portals designed for a targeted group of users and which contain search tools that recognize annotated documents would serve to make other users aware of those documents. Web portals could provide a way to not only compare the annotation of new documents to a set of valid annotations, but also to publicize the annotated documents to the rest of the community utilizing that portal.

Research projects have resulted in the development of many ontologies that contributed to the current proposed designs for the Semantic Web, but only a limited number of documents containing semantic markups have been published. There is an obvious need for ontology-management tools and a number of these have been developed and are being enhanced to keep pace with changing standards. In addition, work is being done to produce layered approaches on the server side to simplify the problem of semantic markups. The lack of true Semantic Web annotated pages is seen as a major drawback for many researchers. Haustein and Pleumann (2002) comment on the severity of this problem and point out the need for more advanced Semantic Web tools that make annotation a simple task for those unfamiliar with the underlying artificial intelligence (AI) concepts. Tools have been developed to verify markups according to particular syntactic rules, but this does not address the more fundamental problem of the semantic quality of markups designed for the Semantic Web. This dissertation was designed to simultaneously address the following two problems: create a way for novice Semantic Web designers to evaluate the semantic quality of the markup within a specific ontology and at the same time outline a method to find unexpected commonalities of terminology used within that ontology.

The Semantic Web design is intended to provide knowledge about many kinds of Web-based resources. A few examples of resources that could be semantically enhanced are calendars, inventory systems, multimedia documents, and repositories of ongoing research efforts. At least six use cases have been published for OWL and the latest variations can be found through a search of the W3C site. At the time of this writing, the use cases are published at http://www.w3.org/TR/webont-req/ on a page entitled "OWL Web Ontology Language Use Cases and Requirements."

The first of these cases describes how Web portals, which are Web sites that provide content information on a common topic, can be enhanced with ontologies (Heflin, 2003). Examples are given for existing Web portals. This dissertation developed a further possible enhancement of these portals, not suggested by Heflin. That enhancement was to define an algorithm to evaluate the similarity of new document annotation effectiveness to a set of validly annotated documents housed within the portal designed for the benefit of those new to semantic annotation to provide a means of semantically evaluating the validity of the document annotation.

Barriers and Issues

The principal issue considered was that the Semantic Web is a project still under development. Further refinements to the proposed design are to be expected. The publication of requirements for OWL is one example of this (W3C, 2003). Incorporating the design criteria, as it became available, added to the complexity of this study.

The amount of research in the area of knowledge discovery applied to semistructured data is continually expanding. The evolving standards for knowledge representation, and specifically for a Semantic Web, have developed over a short time period. The relative newness of this field has precluded in-depth analysis of the effects of this type of data structure on knowledge discovery, as full adaptation has not yet occurred.

At the current time, the OWL language has been adopted by the W3C as a semantic markup language designed to enable sharing and publishing ontologies on the Web. The OWL language, an extension of RDF, has grown from previous work. It builds on RDF and the RDF Schema (RDFS) by adding more vocabulary to describe classes, properties and relations. Utilizing both OWL and its predecessor languages, work has begun into researching how web pages can be annotated through the use of prototype ontologies. Due to the newness of this field, there is not currently an extensive set of publicly accessible Web pages based on a specific annotation scheme. One set of documents that does exist and is publicly available is those abstracts of papers presented at the International Semantic Web Conferences. The annotated conference documents published for the years 2002-2004 were analyzed for this study. The analysis performed for this dissertation encompassed data from the ontology developed for that conference annotated according to two different formats as the techniques have developed over these three years included in this dissertation.

Limitations

Although data currently available was valid in terms of analyzing algorithms, it was not extensive enough to allow in-depth analysis of the scalability concerns associated with these algorithms. The lack of available data for research studies has been noted by many authors including Anyanwu, Maduko and Sheth (2005) and Haustein and Pleumann (2002). For this reason, some aspects of the topic of scalability are suggested as an area for future research. Modifications to the tool developed for this analysis designed to address increased performance are also suggested for future research.

Current Semantic Web ontologies contain very broad categories of information that can be annotated into the document. Different authors select different focal points based on the intent of the document being produced. For that reason, not every document annotated with the same ontology may have annotations designed for the same purpose. Criteria were incorporated into the tool developed for this study to select those documents deemed most appropriate to the similarity analysis being undertaken, based on the ontological similarity as correlated with the text similarity.

W3C working groups continue to refine the OWL language and develop standards for OWL-based web services. Since much of this work is still in development, this dissertation provided an analysis of similarity measures in existing documents, but left development of actual tools to use the results of that research to be done as the standards are more fully developed. Due to the changing standards, documents using both OWL-based annotations and annotations developed prior to the OWL standards were utilized.

Summary

Information is an important commodity in today's marketplace. There has been research into ways to glean information from other Web sites. Liu, Ma and Yu (2001) show a method to discover terms on remote Web sites with a profile entered by the user. Maedche and Staab (2000) show how to use ontological information contained in relationships between concepts to aid in the processing of text. One aim of this dissertation was to extend the concept of the work of those and other researchers in order to present a method for discovery of unexpected common terms based on an the concepts contained in an ontological profile as opposed to one entered by the user. Another goal was to evaluate the correlation between the similarity measure and the common terms in document pairs various based on measures of similarity. This involved parsing the ontology to identify relationships that apply to the specified high-level concepts of the ontology and querying the annotated documents to find the objects of those relationships. Analysis of the effect of unexpected knowledge was accomplished by utilizing a comparison of similarity to all terms and to only those terms flagged as unexpected. Document selection was done by determining those documents with the best fit to the overall analysis. A comparison was made of various measures of document text similarity. Through a process of successive removal of word suffixes, terms were reduced to roots and terms containing the same root are combined for the purpose of the analysis. A stop-list, which is a list of common non-descriptive terms, was used to also eliminate terminology not specific to that document. From the reduced list of terms in each document, comparisons were done to find and determine if common terms are more likely in documents with a higher degree of ontological similarity.

Part of the research for this dissertation involved parametric studies that analyzed a set of documents annotated for the Semantic Web. The annotations in the documents were compared to the text of the documents. A subset of the documents was selected for use in the final similarity analysis, which consisted of those documents having the highest correlation of common terms to a base similarity measure. The parameters that were varied include the choice of which of the highest-level concepts in the ontology to consider in the similarity analysis, whether the correlation will be based on common term or common unexpected terms, whether or not those documents having no common terms were considered, and the criteria for selecting the best subset of the documents to use in the similarity analysis. A parameter involving the selection criteria was the threshold value used to determine which documents should be deselected due to the value of the difference between rankings of the common terms and the similarity estimate. Original plans had been to vary the maximum number of times any particular document may be deselected before it was removed from consideration with this set of parameters, but actual analysis showed that a steady-state was quickly reached, where no further improvement was possible, so this parameter remained constant for all data considered in the final analysis. A method to determine which documents are most appropriate for use in similarity comparisons was proposed. Various ways to measure the similarity of the annotations were explored. Conventional measures of similarity were considered, which include the simple match technique and the Jacquard measure (Losee, 1998), and the vector space or cosine measure (Liu, Ma and Yu, 2001). Also considered were enhanced measures that utilized the hierarchical relationship of ontological terms in the Semantic Web as proposed by Ganesan, Garcia-Molina and Widom (2003) and two measures proposed in this dissertation that combined both traditional and hierarchical measurement techniques. Terms in the documents were grouped according to term roots, which were derived by elimination of common suffixes from the words., and correlations were calculated between common terms among document pairs and the various ontological similarity measures. This was presented as the basis for future work to develop a tool to provide semantic validation of ontological markups.

The use of portal-based tools for the Semantic Web has been previously addressed in the OntoEdit project (Sure, Erdman, Angele, Staab, Studer, & Wenke, 2002), by providing a collaborative environment for researchers to advance their work, as opposed to novices to the Semantic Web environment. The algorithm presented in this dissertation would follow the principles of collaborative development that were found to be effective by those researchers, but would differ in that the targeted users would be less experienced in design of SWDs. Despite the variety in the types of documents that exist, many documents on today's Web are primarily textual and intended to be understood by a human reading the document. One aim of the Semantic Web effort is to define a way that all Web resources can be enhanced to enable agent-based processing. The dissertation focused on one specific class of documents (text-based documents) that are often produced by novice Web authors. The analysis done in this simulation was designed to lay the groundwork for development of a tool to be used on Web portals, which would allow novice Semantic Web designers to compare a proposed ontological markup to a set of documents that have been selected for the validity of the ontological annotation as compared to the text of the document. This approach is particularly well suited to a web portal environment, since the intent of a portal is for use by those with a common identified interest such as those annotating documents using the same ontology. In addition to providing access to the analysis software, the portal Web site could provide a way to access both the ontology for that set of users and the set of validated annotated documents.

As the transformation of all businesses to e-businesses continues, there will be a continued demand for techniques to delve deeper and find more meaning in the data available on the Web. An organized procedure to allow Web authors to determine if the annotation they have provided defines their Web page in a way similar to other Web authors using this annotation could be beneficial in producing semantically annotated documents that can be confidently employed by a community of users. A possible first step in this process is to define methods for similarity comparison and extraction of unexpected common terms and to analyze the results of utilizing those methods with actual data from publicly accessible semantically annotated documents. This will provide a framework for comparison that can be applied as the syntactic definition of web ontology languages continues to evolve.

The research of literature shows the existence of many methods of similarity computation and that these methods have been applied to Web documents. Previous research has been done to apply clustering techniques, analysis of vector spaces, hierarchical structures, and profiles entered by the user. In most previous work, either the documents themselves were analyzed to determine similarity, the Web page structure was used to determine a measure of similarity, or the metadata in the page was used. The research in this dissertation built on and combined several of these methods. Rather than a simple comparison of keywords, the approach here combined selected keyword metadata that could be extracted from the ontological markup with other data that could be inferred from the structure of the ontology. The horizontal relationships that were implied through the structure were used to extract the original metadata and the hierarchical arrangement of that data within the ontology was used to infer additional information. Various similarity measures were adapted for use in this context and these measures were compared to determine which are most effective in measuring the similarity based on the ontological annotation.

The primary goal of this dissertation was to determine a method of calculating valid and appropriate measures of similarity of ontological notations. This was done by an examination of annotated documents to determine if a correlation existed between the ontological similarity and terms contained in the document pairs. Other goals were to present an overview of the ontological development for the Semantic Web project and to demonstrate a way to apply accepted techniques for discovery of common terms to textual data (Web pages) based on the Semantic Web annotation. With the realization that the Semantic Web is a work in progress, this dissertation aimed to show a rational basis and feasibility of the tasks and defer the actual development of tools to accomplish this on a large scale for future study as the Semantic Web syntax and constraints are further defined. A software tool developed to calculate various measures of similarity, to extract common terms in document pair and to calculate the correlation between similarity and common unexpected terms was provided. This tool handled selection of those documents most appropriate for this type of analysis from a larger set of annotated documents.

Chapter 2

Review of the Literature

Evolution of the Semantic Web

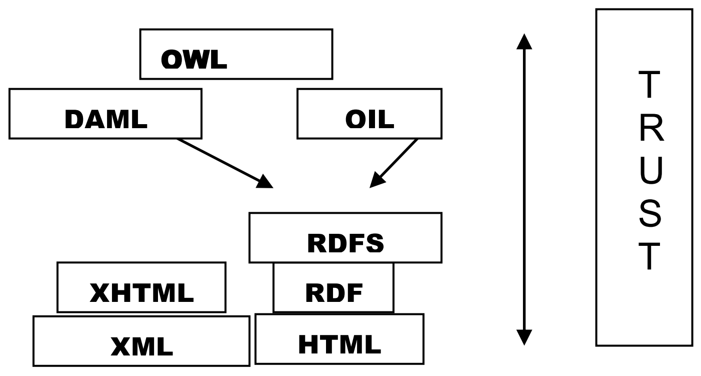
The current Semantic Web effort evolved from previous and current work at many institutions and in many fields. The language of the World Wide Web (WWW), Hypertext Markup Language (HTML), is a language designed for the visual presentation of information for humans to view. The current proposals for a Semantic Web language are based on a layering of languages built up from HTML. Extensible Markup Language (XML) provides syntax to encode information about that viewable data into a Web page. It gives a way to make the data available for other tasks. XML provides a foundation on which prototype languages for the Semantic Web have been built. Goldman (2000) showed how the research concerning semistructured data could be applied to XML to create query tools for Web-based data. Heflin (2001) described an XML-based ontological markup language for Web documents. Heflin's work showed how the Semantic Web proposals developed beyond the elements and document type definitions (DTDs) of XML. Prototypical Semantic Web languages are based on the theoretical foundations of knowledge representation (KR) including semantic networks, frame systems, descriptive logics, predicate calculus or first-order logic or F-Logic (Kifer, Lausen, & Wu, 1995), ontologies, context logic, deductive databases and distributed databases.

In addition to developing languages for the Semantic Web, there has been much collaborative work in other areas. Academic institutions, government-sponsored institutions, and private industry have joined in many initiatives. Two government-sponsored agencies have been especially active in the development of a markup language to be utilized by Web agents. In October 2002, a joint committee was created to coordinate the work of the Information Society Technologies Program (IST) of the European Union and the Defense Advanced Research Projects Agency (DARPA) in the United States. As one result of this joint committee, many research projects have been jointly developed and published by those two agencies and universities and industrial researchers supported by those agencies.

On a very simple plane, the current work concerning adding meaning to the data presented on the Web can be viewed as a way of organizing metadata, or data about data. Ontologies allow inferences beyond simple keyword metadata. Much of the research done in developing standards for metadata has been applied to the development of ontological standards. The Dublin Core Metadata Initiative (DCMI) is an interdisciplinary, international group founded in 1994 dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems. The Dublin Core Metadata Element Set provided a set of fifteen elements that are useful in describing almost any published data. On the Web, this could be accomplished via the HTML <meta> tag. The problem with this type of metadata is that it is ambiguous, since the meaning of a term can vary from one Web page to another. XML provided a means to avoid this problem by using a specific namespace. The concept of namespaces was core to the development of a truly unambiguous web language, as terms have different meanings in different contexts. In addition, the Resource Description Framework (RDF) provided a basic structure to link concepts together to express meaning. In October 2002, DCMI issued a recommendation that is explained in the document "Expressing Simple Dublin Core in RDF/XML" (Beckett, Miller, & Brickley, 2002). DMCI plans to issue a series of recommendations, beginning with this one, to encode Dublin Core metadata in a way that conforms to other developing Web standards (Dublin Core Metadata Initiative, 2002).

In Web terminology, a resource is an address on the web. A Uniform Resource Locator (URL) points to a specific web page address. A universal resource identifier (URI) is more general and points to any resource on the web. One web page that can be retrieved by a single URL may provide multiple URIs. Tim Berners-Lee (W3C, 2003) envisioned the new Semantic Web to be a “Web of Trust”, with authorities to authenticate the resources being referenced. This is to be accomplished through a layering of existing technologies enhanced with authentication. An adaptation of the building blocks he had envisioned for the Semantic Web is shown in -. Many depictions of the layers of the Semantic Web show trust as an uppermost layer, and the components of encryption and signatures as being pervasive throughout upper layers. An example of this depiction can be seen at:

http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-2.html.



Legend

DAML:	DARPA Agent Markup Language
DARPA:	Defense Advanced Research Projects Agency
HTML:	Hypertext Markup Language
OIL:	Ontology Inference Layer or Ontology Interchange Language
OWL:	Web Ontology Language
RDF:	Resource Description Framework
RDFS:	Resource Description Framework Schema
XHTML:	Extensible Hypertext Markup Language
XML:	Extensible Markup Language

Figure 1: Envisioned Layers of the Semantic Web

HTML was designed as a way to present data on the Web. Tags for HTML are meant to express how the data should be visually expressed, not the meaning of the data. XML is meant to better define the meaning of data and can limit that meaning to a specific namespace, thus allowing the same descriptor to have different meanings in different contexts. XML allows the user to define tags to describe the data and optionally to define a descriptor for the tags. According to the charter for the working group tasked with development of a Web Ontology language, HTML and XML were the foundation on which future Web enhancements must build (W3C, 2003).

RDF is another building block in the design of an ontology language for the Semantic Web. RDF provides one methodology to design architecture for metadata. In RDF, each concept is expressed as triple, or a combination of three basic ideas. Each RDF triple has a resource (or subject), property (or predicate) and object (or value). The resource is designated by a URI. RDF provided a means to link together metadata in a way that can be understood by machine agents. While the basic idea of an RDF triple is simple and straightforward, it can also be limiting in the expression of data. The English language provides adverbs to modify the predicate in a sentence, but in RDF the modification of the predicate of a triple is more complex. For example, the simple statement "Bill painted the fence" can be broken down as subject (Bill), predicate (painted) and object (fence). However, if the statement is enhanced with how the fence was painted, then the problem becomes more complex. The new statement "Bill painted the fence with a brush" does not directly translate into a single triple. In order to express the full idea, it is necessary to change the predicate (painted) to become a new object (the painting event). This process is known as reification. RDF can be viewed as an expression of directed graphs, where the subject and object are nodes connected by the predicate, which is represented as an arc. Reification can be defined as the translation of an arc to a node in a graph. RDF allows modifiers for the nodes (subject, object) but not the arcs (predicate). Providing a way to fully express the meaning of data and keeping that expression within the confines of XML adds to the complexity of RDF.

Much research done by many different people has gone into the development of tools to create and parse RDF documents. A compilation of many of these RDF tools was developed by Dave Beckett and available online at http://planetrdf.com/guide/. Tools, such as Protégé (Protégé Project, 2002), have been developed provide a graphical user interface (GUI) to assist in the building of an ontology. Libraries of existing ontologies have been compiled. Annotation tools have been developed to assist with using ontologies to enhance documents. Visualization tools show the relationships between classes, as well as the graphical depiction of the RDF nodes and arcs. One visualization tool, called OntoKick, was developed as part of the OntoEdit project (Sure et al., 2002). Validation tools include theorem provers, programs to check for class consistency and applications to verify syntactic correctness. Inference engines deduce relationships between the ontological definitions and are able to handle semantic queries. Several of the major Semantic Web projects have compiled libraries of toolsets for RDF and languages evolved from RDF. The the World Wide Web Consortium (W3C) has set up a repository of various semantic web tools at http://esw.w3.org/topic/SemanticWebTools.

RDF lacks any type of data typing mechanism. This can be provided through the Resource Description Framework Schema (RDFS). The schema extends RDF allowing attributes to further describe the resources, and allows the definition of vocabularies, structures and constraints through the extension of the modeling primitives in RDF. Semantics can be defined through relationships such as class/subclass, which are not a part of the RDF specification. The RDF Schema gives a way to provide information concerning the meaning of the statements in an RDF data model, but does not limit the syntax of the RDF specification (Broekstra, Klein, Decker, Fensel, van Harmeien, & Horrocks, 2001).

Modeling primitives in RDFS contain core classes and properties. Collections are supported in the schema. The original core classes were resources, properties and classes and now include literals, data types and XML literals as well. Concepts in the schema are of the class datatype, which is specified by rdfs:Class. An RDF statement, specified by rdf:Statement, is the statement made by an RDF triple, which consists of a subject, predicate and object. The subject and objects are usually identified by instances of an rdfs:Resource, while the predicate of an RDF statement is the instance of rdfs:Property. The current specification proposal is available online and review of that document recently concluded (Brickley & Guha, 2003).

One major project of the European-sponsored IST was known as On-To-Knowledge. One result of this project was the development of a proposed Ontology Inference Layer, or Ontology Interchange Language (OIL). This was the first language based on the proposals of World Wide Web Consortium (W3C) (Welcome to Ontoknowledge, 2002). OIL was designed to bring together several theoretical premises into a practical framework, combining the semantics and reasoning support from Descriptive Logics (DL), the modeling primitives from frame-based systems and the Web standards proposed by the W3C. According to Broekstra, Klein, Decker and Fensel

(2001), OIL implements a well-defined first-order semantics and provides automatic reasoning support.

A similar project was American-sponsored DARPA Agent Markup Language (DAML), which was funded by DARPA (Fensel 2000). DAML is an XML-based semantic language and is intended to lead to the design of ontologies that are consistent with Web standards, are reusable and can be easily extended. The project was designed to not only define the language, but also provide a toolset for implementation of that language.

Though similar in intent and design, there are some differences between DAML and OIL (Fensel 2000). DAML actually inherits many aspects from OIL. Both languages: support hierarchies of classes and properties; allow building of classes through the use of intersections, unions and complements; allow restrictions of domains, ranges and cardinality; support both transitive and inverse properties; and support data types such as integers (Fensel 2000). DAML does not provide the degree of backward compatibility with RDFS that OIL does. OIL can state conditions for a class that are necessary, sufficient, or both. This allows for a degree of automatic classification of classes within OIL.

DAML+OIL was a proposed Semantic markup language for Web resources that combined features of both language specifications. It added more modeling primitives to the existing specifications for RDF and RDFS, and provided primitives similar to those found in frame-based languages. DAML+OIL enhanced the DAML semantics with the OIL inferencing capabilities (Fensel 2000).

A new language for producing ontologies on the Web, known as OWL, has been proposed by the W3C. OWL is a direct successor of the DAML+OIL project of the Joint US/EU ad hoc Agent Markup Language Committee and much of the DAML+OIL was derived from previous work on DAML and OIL. In February 2003, a working draft of the abstract semantics and syntax of this new language was posted. A formal definition of the language is provided through the proposed model-theoretic semantics. OWL is an extension of the RDF model (W3C, 2003).

Three versions of the language are described. OWL FULL contains all OWL language constructs and allows full use of RDF along with OWL. Two sublanguages are proposed, OWL Descriptive Logic (OWL DL) and OWL Lite. OWL DL imposes constraints on some of the language constructs. Classes, datatypes, data and object properties are required to be distinct. Object properties are distinct from datatype properties. This means that the same entity cannot be a class and an individual in OWL DL. OWL DL requires that all classes and properties be explicitly defined in the same OWL ontology. OWL Lite builds on the restrictions of OWL DL, adds further restrictions to the syntax and attempts to provide a minimum set of useful language features (W3C, 2003).

In addition to the working draft for OWL abstract syntax, a document outlining the requirements and a set of use cases was also posted. The primary goals of OWL include interoperability, detection of inconsistencies, standards compatibility, internationalization, scalability and ease of use. Use cases depict how web portals, multimedia collections, corporate web site management, design documentation, agents, services, and ubiquitous computing can be improved through Web ontologies using

OWL. Requirements for OWL include that the language should incorporate the customized tagging scheme of XML and utilize the flexible approach of data representation provided through RDF (Heflin, 2003).

As the languages of the Semantic Web continue to become more refined, there is a need to insure that the languages can be understood in a consistent and logical way. The RDF working group of the W3C has proposed that semantics be defined to create a "base language" of the languages developed for the Semantic Web. This language, tentatively named Lbase, would provide a way to map from one language to another. In January 2003, the working notes for the group developing this language were posted online by Guha and Hayes at http://www.w3.org/TR/lbase (W3C, 2003). At the time of this writing, the 2003 document is the latest one concerning Lbase on the W3C site. This document does mention limitations of Lbase, some of which are addressed in the OWL language and was adopted after the Lbase posting. Research has continued into ways to map ontologies together that were developed according to various standards.

Semantic Web Projects

An excellent explanation of the basic principles of ontological expression can be found in the dissertation by Jeffrey Heflin (2001) while working on the Simple Hypertext Ontology Extensions (SHOE). This document explained the dual development of web ontologies from both the standpoint of first order logic and from frames. In addition, he described the SHOE projects and the contributions the work documented by his dissertation made to that project. Although the SHOE extensions do not conform to the current RDF-standards, SHOE served as a prototype language for an ontological markup language for the Web. This project, centered at the University of Maryland, has now evolved into other research at that same university (Heflin, 2001). The new project is now known as the Semantic Web Agents Project (SWAP), and is a part of the MindSwap initiative at the University of Maryland. (http://www.mindswap.org, 2003).

Another series of projects that have had significant impact on the Semantic Web effort have been centered at Stanford University. The Lightweight Object Repository (LORE) project, which was completed in 2000, provided a foundation for semistructured database management systems. This system was tailored for XML and led to the development of query and search tools for XML databases. In his dissertation, Goldman (2000) describes algorithms used in that system for effective searching of XML data and combined querying of traditional and Web-based data.

Protégé is a tool developed at Stanford that allows users to construct and maintain domain ontologies. The system provides a platform to allow access to other knowledge-based systems. A recent addition to the Protégé project is a version that has a back-end for DAML+OIL support (Protégé Project, 2002).

Another tool developed at Stanford is Chimaera (McGuinness, Fikes, Rice, & Wilder, 2000). This is a software system designed to allow users to build and update large ontologies in a distributed environment through the Web. This was a follow-on project to Ontolingua (McGuinness, Fikes, Rice, & Wilder, 2000), which provided a Web-based server for collaborative, distributive development of ontologies, authoring tools and a library of reusable ontologies. Chimaera was designed as part of DARPA's High Performance Knowledge Base (HPKB) program to provide tools to support merging of ontologies from various sources, evaluation of the correctness of ontologies and the issues that arise from ontology maintenance. The project addressed constraint violations that occurred when ontologies were merged, design of tools to maintain the completeness of taxonomies from the merged ontologies and changes necessary to insure ontologies are reusable. This project differed from some other ontology development efforts in that the focus was on maintaining consistency rather than extending reasoning capabilities (McGuinness, Fikes, Rice, & Wilder, 2000).

Members of the Stanford University Database Group, along with personnel from the Framework for Distributed Organizational Memories (FRODO) project supported by the German Ministry for Education and Research, were instrumental in the development of Triple. This is an RDF query, inference and transformation language designed for the Semantic Web. This language, based on Horn logic, contains Frame Logic, also known as F-Logic (Kifer, Lausen, & Wu, 1995), which provides a logical framework for object-orientated languages integration with databases and knowledge representation. The FRODO language is designed for querying and transforming RDF models. The language employs a layered approach feature definition for specific data models. One kind of layer that is supported allows basic RDF constructs such as resources and statements to be included. Another type of layer directly supports existing modules for semantic expression, such as DAML+OIL (Sintek & Decker, 2002).

WWW was initially a creation of Tim Berners-Lee and was developed at the European Council for Nuclear Research, which is the English translation of the French name Conseil Européen pour la Recherche Nucléaire (CERN) in Switzerland. It has continued to be a major focus of research at several European institutions. The IST-sponsored On-To-Knowledge project was designed to make knowledge management easier for those non-IT specialists working in the knowledge management field. The focus was to develop processes and software applications that would assist those workers in utilizing an ontological approach to manage the Web-based information that was their responsibility. OIL was one of the early products of this project. Six companies based in the Netherlands, Switzerland and Germany teamed with the University of Karlsruhe in Germany and Free University in Amsterdam. Knowledge management techniques were the focus of the research at the German university, while ontology based modeling was the focus at Free University.

Strong links from previous collaborations already existed between these academic and commercial partners. During the time the project was active, from 1999 until 2002, over forty deliverable products were produced. These included the OIL language itself, various ontology tools including those for interoperability and scalability and tools for collaboration. This project led to a number of continuing follow-on endeavors sponsored by IST that include Onto-Web, SWAD-Europe, WonderWeb and many others (Welcome to Ontoknowledge, 2002).

OntoEdit, a part of the On-To-Knowledge project, was developed as an ontology editor that brought together many different ontology engineering methods. OntoEdit was one of the tools used in the collaborative development of an ontology to support distributed research via a Web portal. Maedche (2002) described OntoEdit as an ontology engineering workbench, providing an import wrapper, importer, merge tool, ontology crawler, natural language processing system, document wrapper and transformation module. Algorithms for ontology extraction and maintenance were stored in the library of this project.

The SEmantic portAL (SEAL) project (Sure et al., 2002) provided a conceptual framework for the portal development to support Semantic Web research at the University of Karlsruhe. This project aimed at taking advantage of semantic structures to present information to human and software agents. A knowledge warehouse and an inferencing mechanism formed the backbone of the system. Within the knowledge warehouse were the ontology, or general logical theory and the knowledge base that described specific circumstances. The inferencing mechanism for SEAL was Ontobroker, which functioned like a middleware run-time system. Software agents, community users and general users made up the front end of the system and they interacted, via the web server, with various applications such as the RDF generator, query module and input/output applications that collected and dispersed data via forms and templates. One component of this project was that of semantic ranking. In this project, similarity of two knowledge bases was reduced to the similarity of concept pairs, based on F-Logic (Maedche, Staab, Studer, Sure, & Volz, 2002).

Using SEAL, a collaborative ontology development project was undertaken at the University of Karlsruhe via a Web portal to describe the institute doing this research. During the requirements phase, the requirements specification for the ontology was set up by the ontology engineers. Collaboration extended to other related research groups and domain experts. Two tools were used in this phase. OntoKick (Sure et al., 2002) was used to assist in the creation of the requirements specification document and to pull out the structures needed to describe the ontology. OntoKick allowed interaction between the team members when determining goals, guidelines, targeted users, use cases and other top-level criteria. OntoKick provided a format for competency questions that defined queries the ontology should be able to answer. The second tool, Mind2Onto (Sure et al., 2002), supported brainstorming by creating mind maps of the discussion information. In the refinement phase, a transaction management protocol was used to enforce locks needed when multiple designers are working on the same database. In the evaluation phase, OntoEdit (Sure et al, 2002) provided tools to edit instances and axioms to create query test sets, graphical editors to assist with error location and avoidance and use competency questions for evaluation.

Semantic Web Advanced Development in Europe (SWAD-Europe) was designed to provide Semantic Web research, show ways that the Semantic Web could be used and reach out to the larger community. The project is designed to assist those in the field of networked computing to fully incorporate Semantic Web technologies. Brickley, Buswell, Matthews, Miller, Reynolds and Wilson (2002) explained the aims of this project, which grew from the On-To-Knowledge project. The project, which began in May 2002 and was funded through 2004, had five major goals. Included in those goals was implementation of example scenarios that demonstrate the integration of multiple Semantic Web technologies as well as development of a technology integration strategy. Also included among the goals was to ensure the European community was kept aware of accountability, device independence and internationalization that the Semantic Web could provide. Another goal was maintaining awareness of international best practices and ensuring international awareness of European best practices. The undertaking of research and development designed to support these objectives was the final goal. The project concentrated on addressing the need for convergence of services for the Semantic Web and integration of RDF and Web ontology languages with XML specifications.

Other areas addressed by the project were coordinating query language advances and application program interfaces (APIs) to support them, researching the management of trust and authentication issues, identification of commonalities and best features of annotation tools, scalability and management of thesaurus systems. The five partners were the W3C, the research team at the University of Bristol and three industry partners.

Ontology Research for the Semantic Web

Much work in recent years has been focused on the development of ontologies for many areas of study that conform to the overall Semantic Web design. Most of these ontologies, based on the RDFS standards, include some type of inferencing capability. In most cases, the inference engine is provided using tools or languages employing DAML + OIL or subsequent languages.

One area where there have been several efforts to create a usable ontology is the area of academic research. There are specific requirements in most peer-reviewed publications and these requirements can be used to form a set of knowledge that can be encoded into documents to facilitate information retrieval. Although this is an area of current development, there are some examples of repositories of Web pages created with a specific ontology. Lopatenko (2001) presents a comparison of several efforts at creating research ontologies, as well as the specifics for the Current Research Information System (CRIS). The International Semantic Web Conference (ISWC) 2002 (International Semantic Web Conference 2002 Web Page, 2002) required all submitted papers to have an abstract with annotations utilizing the iswc.daml ontology. Papers were also submitted to the organization’s next two annual conferences with annotations from that ontology and the subsequent one based on the OWL standards.

Semantic Web tools depend on the development of some type of standardized classification scheme. Various working groups in the W3C have developed ontological structures that can be used to annotate documents in a format consistent with the latest proposals for the Semantic Web. One such group has developed ontology to be used with research about the Semantic Web. This effort has become known as the “Semantic Web Research Community Ontology” (SWRC Ontology). It provides a way to encode semantic information about researchers, sponsoring organizations and other bibliographic metadata (Sure, 2001). The original Semantic Web working group has evolved into the OntoWare Group, which provides project support and software products at no cost to interested Semantic Web researchers. The Semantic Web working group has examples of Web pages annotated with this ontology available on the OntoWare web site at http://ontoware.org/projects/swrc/.

Another ontology was developed for academic research as part of the Scholarly Ontologies (ScholOnto) project. This project augments Web documents with the readers’ assessments of the claims in the document. A claims server system, named ClaiMaker (Li, Uren, Motta, Shum, & Domingue, 2002), is used to track the information. Query tools allow interpretations across more than one document. This project provided another example of another Web-based tool aimed directly as a specific community of Semantic Web users.

Semistructured Data Models

It is important that the developing ontologies be able to incorporate previous work in the fields of data modeling and knowledge discovery. The current work in the Semantic Web can trace its roots in a number of directions. XML, one of the building blocks of Semantic Web languages, can be seen as a language to enforce a semistructured data model on the web. A brief explanation of semistructured data models is included here to aid in understanding how some ontological development has come from this area.

Most database research in the last few decades has centered on relational database management systems. These systems assume the data will conform to a rigid schema, but this is not true for all sets of data. While some data is truly unstructured, a large class of data can be fit into the category of semistructured data. Various authors define this term differently. In their article, "Discovering structural association of semistructured data" (2000), Wang and Liu give the following explanation:

> Semistructured data arises when the source does not impose a rigid structure (such as the Web) and when data is combined from several heterogeneous sources (such as data warehousing). Unlike unstructured raw data (such as image and sound), semistructured data does have some structure.

Peter Buneman, in his 1997 article "Semistructured data", commented that semistructured data is often described as being "self-describing". He went on to say that the data might be of a form that cannot be described by a schema, or it may conform to a schema that only places loose constraints on the data. Since Web-based data fits in the broader category of semistructured data, research on semistructured data has been useful in developing algorithms for the Semantic Web.

The interest in semistructured data has risen dramatically in the last few years, as XML has become a standard way to store semistructured data in Web documents. The rise in data warehousing applications also created an interest in developing techniques to organize data from various disparate schemas. There is current research into the theoretical basis for processing of semistructured data, as well as developing working systems to manage this type of data.

Various algorithms have been developed to assist with the management of semistructured data. In addition, some database management engines have been written expressly to deal with semistructured data. In order to understand the current methodologies for processing semistructured data, it is necessary to understand how this data is usually represented.

A structured database with null values replacing those not present in a specific data item may be used to represent semistructured data, but other approaches have also been suggested. Structured data can be represented in a two-dimensional table, but it can also be represented as a tree, with each row of the table shown as one leaf from the root. In the case of object-oriented data, the tree-model is not sufficient, as objects can point to each other (Abiteboul, Buneman, & Suciu, 2000).

Much of the research in the field of semistructured data has utilized the Object Exchange Model (OEM) as a way of representing the data description (Papakonstantinou, Garcia-Molina & Widom, 1995). Another standard representation of data has been the object-oriented data model specified by the Object Data Management Group (ODMG). The outline of a structure that may be present in XML documents has also been important to understand in this research field. In their book, *Data on the Web: From Relations to*

Semistructured Data and XML (2000), Abiteboul, Buneman and Suciu presented a grammar for semistructured data and demonstrated how it compared to the XML standard. A brief overview of these topics is a necessary preface to in-depth discussion of current research.

OEM was originally designed to exchange data between heterogeneous sources in the Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) system developed at Stanford University (Garcia-Molina et al., 1997). Another related project at Stanford, the LORE system (Goldman, 2000), expanded this model to include edge-labeled graphs. This version has become the standard for modeling semistructured data. An OEM object contains a label, object id (oid), type and value. If the type is complex rather than atomic, the value is a set or list of oids. If the value is not complex, then the value is represented by an atomic value such as a numeric value or a string. The original OEM model was a graph where the labels were attached to the nodes of the graph. A variant of that model which has been widely used attaches the label to edges of the graph rather than nodes (Abiteboul et al., 2000).

The ODMG specification formed the standard of object databases (Abiteboul et al., 2000). An Object Definition Language (ODL) was used to express the schema. Constructs common to most object database management systems, which are supported by the ODMG model, include:

- Built-in tuple and collection types including set, list, multiset and array types were supported.
- Persistent roots (or handles) to the objects in the database were provided.

- Language bindings determined the style of destruction of objects, either explicit or through garbage collection techniques.

The standards provided adequate flexibility to allow description of both structured and semistructured data. The standards included an informal specification for an Object Interchange Format (OIF), which allowed a way to textually format the data (Abiteboul et al., 2000).

In the dissertation *A Semantic Paradigm for Intelligent Data Access,* Vaschillo (2000) discussed the use of the Open DataBase Connectivity (ODBC) Standard as a data model to provide an interface to the WWW. In this model, Web documents were viewed as semantic objects. Using this model, the author presented a semantic database management system that could be used to process queries across the Web.

The W3C has adopted XML as a standard to complement HTML for data exchange on the Web. While HMTL tags are designed to specify the presentation format of a web page, XML is meant to describe the content. It differs from HTML in that new tags may be defined at will, structures can be nested to any depth and the document can contain a document type definition (DTD). Since the DTD is not required for XML documents, it cannot be relied on exclusively as a schema-definition tool, but can be exploited in documents where it is present. Each element in an XML document is bounded by matching tags, which can surround raw text, other elements, or a combination of the two. XML allows attributes (or properties) to be associated with elements. Attributes are defined with name, value pairs. The same information could usually be displayed as nested XML tags. Abiteboul, Buneman and Suciu (2000) give the following example. Information about a person named Alan could be stored as nested elements.

```
<person> <name> Alan </name>
         <age> 42 </age>
</person>
```

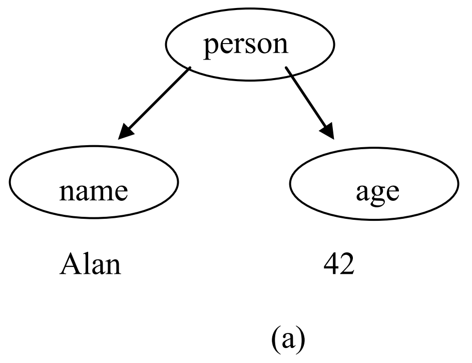
This same information could be conveyed as an attribute of the person.

```
<person name = “Alan” age= 42 />
```

This attribute form could easily be rewritten in a more common way to specifically denote semistructured data expressions (ssd-expressions) as

```
{person : {name: “Alan”, age: 42}}
```

Figure 2(a) shows the tree-representation of the expression for standard XML and Figure 2(b) demonstrates the tree as a semistructured data expression.



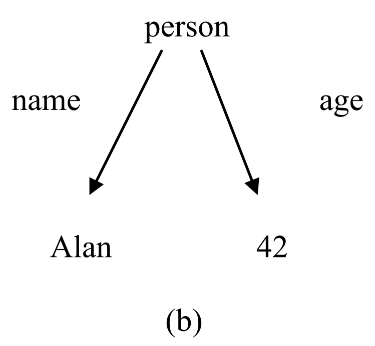


Figure 2: Tree representations XML and semistructured data expressions

There are currently at least two application programming interfaces (APIs) for XML documents. The Simple API for XML (SAX) is a standard for parsing XML data. It is syntax-driven. The interface reads the flow of the XML data, parses it and detects events when interpreting the tags. The events are sent back to the application (Abiteboul et al., 2000).

An alternative API is the Document Object Model (DOM). It provides an object-oriented view of the XML data. DOM defines a Node interface, as well as Element, Attribute and Character-Data subclasses. The DOM API has been adopted by the W3C as the standard for XML data extraction (Abiteboul et al., 2000).

The Simple Object Access Protocol (SOAP) is a standard way to send information in a decentralized distributed environment. The W3C has published a technical report with a working draft of the protocol at http://www.w3.org/TR/soap12-part1/. SOAP is designed to be platform-neutral. It is not dependent on products of any specific vendor. The implementation is in XML and HyperText Transfer Protocol (HTTP). It is easy to implement and messages can be passed across firewalls if the network protocol used is HTTP (W3C, 2003).

Models for semistructured data have been used as a foundation to create metadata. Ontologies, in turn, are based on metadata. Ontologies go far beyond simple metadata by providing relationships and rules to incorporate a way to impart knowledge about the metadata to users. These users could include both humans and machine agents. Ontologies, particularly those that provide for constraints and inferencing, go far beyond standard semistructured models. Ontologies allow rules to be built into systems and provide a means for validation of those rules.

KDD Research

The field of knowledge discovery has become very broad. The design of the Semantic Web is intended to make knowledge about Web resources more usable to agents processing that data. There have been some recent projects aimed at discovering knowledge in Web-based data.

It is important to build on the concepts of previous work. One example of this is shown by Benetti, Beneventano, Bergamaschi, Guerra and Vincini (2002). These researchers addressed the problem of creating a unified view of data from various sources in the design of the Momis Project. Momis stands for Mediator envirOnment for Multiple Information Sources. Momis is a mediator-based system for information extraction and integration that works with both structured and semistructured data. It relies on the metadata of the information sources. The data model used is based on the ODMG specification. Communication is based on the Common Object Request Broker Architecture (CORBA) standard (Object Management Group, 2005) including a common data model, data wrappers and a mediator. Momis used a variation of the ODL language, along with the object query language (OQL). This variation of ODL language could be used to describe heterogeneous schemas of the data sources. Relationships expressing knowledge within and between the schemas can be employed. Synonyms, broader terms and related terms can be specified in this language. The developers of Momis saw this as an answer to the problem faced by e-commerce companies of not having a common ontological description to form the infrastructure that can overcome naming and structural incompatibilities. In addition, the work of these authors provided an understanding of the underlying structured and semistructured data models and languages on which much of the Semantic Web is based (Benetti et al., 2002).

Lin and Pantel (2001) presented an algorithm for classification of text into semantic classes, or concepts. The authors presented a system, which they call Unsupervised Induction of Concepts (UNICON) that contained an algorithm that differs from similarity matrices used in much previous work. Typically, similarity matrix-based algorithms require the user to set a threshold value, focus on concepts rather than words and do not depend on the similarity of words to determine features. The authors' algorithm was based on previous work in automatic thesaurus construction. Words were viewed in their context. The authors utilized a collocation database and a similarity matrix that are available on the Internet at http://www.cs.ualberta.ca/~lindek/demos.htm. These tools produced a binary relationship between words and their modifiers. Collocation was defined by the authors to mean a dependency relationship between two words that occurred more frequently than by assuming the two words were independent. Words entered into the collocation database were viewed as feature vectors. The more features shared by a word, the more likely that the words were similar. After the collocation matrix had been created, the authors' semantic class induction algorithm was used. A clustering algorithm was used to determine groups of words that are most similar. When a word was put in a particular cluster, it was removed from any others. The algorithm computed the centroid of each cluster and merged clusters whose centroids are very similar. The output was a list of clusters. Advantages of this algorithm were that the output was a set of concepts, it could handle a large number of elements, classification could be done on words with few features and previously unknown words could be classified into previous clusters. The algorithm did not perform perfectly and was highly dependent on having a very large set of data to work with.

As mentioned previously, research performed by Liu, Ma and Yu (2001) sought to glean unexpected information from Web pages of competitors. This was done through a comparison to other pages on the Web based on a user-defined profile. This process had as one objective to find unexpected concepts on competitor sites with respect to the user's page. A word analysis was used, with words appearing in the same sentence defined to be concepts. The authors defined a weighting scheme based on the vectors computed from a specific collection of documents, utilizing the index terms or keyword list. Table 1 lists the definitions provided by those authors that are key to understanding the weighting algorithm.

Symbol	**Definition**
P	number of index terms in the collection of documents
k_i	*i*th index term
$\vec{d}_j$	keyword vector representing document j
$w_{i,j}$	weight of term associated with each term of k_i of document d_j, with 0 value indicating term of k_i that does not appear in document d_j
$\vec{q}$	query vector
$Sim\ (d_j,q)$	similarity of $\vec{q}$ and $\vec{d}_j$
f_{ij}	frequency of term k_i in document d_j
tf_{ij}	normalized term frequency of term k_i in document d_j

Table 1: Definitions used by Liu, Ma and Yu

The following equations were used by these authors. Equation 2.1 is a representation of the similarity between the query vector and a specific document, as calculated by the cosine measure. Equation 2.2 describes the weighting scheme used by the authors, which is based on term frequency and inverse document frequency. This equation comes from the work of Salton and Buckley (1988).

Equation 2.1: $sim\ (d_j,q) = \dfrac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \dfrac{\sum_{i=1}^{p} w_{i,j} \times w_{j,q}}{\sqrt{\sum_{i=1}^{p} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{p} w_{i,q}^2}}$

Equation 2.2: $tf_{ij} = \dfrac{f_{i,j}}{\max f_{i,j}}$, where the value is 0 if the term does not appear in that document

Each document was represented by a vector of keyword terms and their associated weights. For each keyword in the document set, the vector contains has a positive value if the keyword is present in that particular document and a zero value otherwise. The authors utilized a weighting scheme proposed by Salton and Buckley (1988) based on the normalized frequency of the terms in each document being compared. Any term that did not appear in the document was assigned a weight of zero. The authors specified five methods of comparing two web sites. Depending on which method was being investigated, the query vector might be comprised of the index terms from the users' web site or from one of the competitor sites. Association rule mining was utilized to find unexpected concepts in the two pages being compared. The Apriori algorithm (Agrawal & Srikant, 1994) was used to discover the concepts in each page, based on the keywords in each sentence. This was done through association miner software that is part of the SMART (System for the Manipulation and Retrieval of Text) system (Salton & McGill, 1983). Keywords were mined from the user page and the competitor pages separately to allow mining to focus on the specific topic of each page. In this analysis, the competitor page is a Web page identified by the user as a page not developed by the user's organization, but containing relevant information. Often, these pages are part of commercial sites engaging in business similar to that of the user. Examples of unexpected information that might be found from competitor pages could be specific products and services that others in the industry are offering. When considering the case of finding unexpected terms in a competitor page c_i with respect to a user page u_j, the unexpected value of each term k was calculated by comparing the normalized term frequencies where $unexpec\ T_{r,i,j}$ was unexpectedness value for term k. The subscripts r, j refer to the rth term in the user's page u_j, and r, i refer to the rth term in the competitor's page c_i. The unexpectedness was calculated as

Equation 2.3:

$$unexp\ T_{r,i,j} = \begin{cases} 1 - \dfrac{tf_{r,j}}{tf_{r,i}} & where\ tf_{r,j} / tf_{r,i} \leq 1 \\ 0 \quad otherwise & \end{cases}$$

The unexpectedness values were computed and ranked according to these values for each competitor page, enabling the analyst to determine a list of unexpected terms. Those terms not in the user page had a value of 1 and terms not appearing in the competitor page had a value of 0 for that page.

A dissertation written by Pluempitiwiriyawej (2001) suggested a slightly different model than the vector space representation. A multi-dimensioned metric space containing a tree-based heuristic algorithm was used to implement the data clustering. A data merge engine, using a hierarchical clustering model, allowed combining similar and overlapping data items from multiple information sources. XML was used as the data model to represent heterogeneous data. Clustering trees were built containing objects that were related. The relationship is determined by an automatic examination of the characteristics of the data, the tree and the domain space.

The primary focus of the research done by Pluempitiwiriyawej (2001) was the development of a data merge engine to be used in a system capable of integrating semistructured data in an XML format from various sources. One aspect of that project was to determine the degree of dissimilarity of two objects. Pluempitiwiriyawej discussed various measures of similarity that can be applied to semistructured data, including Euclidean distance functions and metric space distance functions. Metric-trees, multivantage point trees (MVP-trees) and metric tree indexes (M-trees) were discussed as appropriate structures for evaluation of objects in metric-space. These structures allow for clustering of objects based on the degree of similarity. The trees were built based on a similarity measure of the distance between two documents represented as vectors in n-dimensional space.

This dissertation differed in several significant ways from that of published by Pluempitiwiriyawej in 2001. The data examined by Pluempitiwiriyawej was from non-heterogeneous sources necessitating the data clustering techniques to perform data classification. The ontological structure of the Semantic Web provides a means to accomplish that step, eliminating the need for the tree-based heuristics to cluster the data. Pluempitiwiriyawej focused on the similarity of the data in the documents, while this dissertation was focused on the similarity of the annotations of the document. Since there is a binary evaluation of the annotation of each possible concept, the weighting for frequency of terms in the document vector employed by Pluempitiwiriyawej was not needed in this work.

Document Selection

In any set of data, it is important to filter the data to select those values that are most relevant to the problem being considered. Irrelevant or outlier data is often excluded from further analysis. In the field of information retrieval, documents to be retrieved are often ranked according to their relevance to a specific query. Losee (1998) cited many research studies that indicate the relevance judgments are one of the weakest aspects of the development process. Chen and Karger (2006) pointed out that document relevance is dependent on the needs of the user and that most traditional relevance models try to maximize the number of relevant documents. Bot and Wu (2004) presented a method that utilized a support measure to incrementally build a data set. The method these authors suggested, known as the relevance feedback algorithm (RFA), was based on the assumption that every document could best be characterized by a set of concept terms. The authors defined a document concept as one that has reasonable support among all queries from all relevance assessments of the document. The RFA algorithm began by indexing documents according to a term frequency matrix, after applying stop lists and stemming techniques to eliminate non-descriptive terms and cluster terms from similar roots. The next step collected relevance feedback based on a search query, utilizing a term-document matrix. In the third step, a weighted learning function was used to calculate support values, as new documents were included. Terms were then reclassified based on support values. The last three steps were repeated iteratively at preprogrammed intervals. According to the research done by these authors, the incremental algorithm improved retrieval effectiveness, as measured by standard measures such as precision, recall and others.

Similarity Analysis

There are many ways to evaluate if two documents are similar to each other. The measurement of document similarity has been an area of interest since the earliest document retrieval systems. These may be simple statistical measures, complex calculations based on solving multidimensional systems of equations, or estimates from artificial intelligence tools. There is no one best measure of similarity. Zobel and Moffat (1998) found in their investigation of various similarity metrics for six different applications that the performance of best metric in one instance is worse than the other metrics in another application. This was reiterated by the work of Ganesan, Garcia-Molina and Widom (2003), who pointed out that various measures are more appropriate depending on the situation where they will be used.

Losee (1998) explained a number of factors that must be taken into consideration when evaluating a method of similarity calculation. One commonly used measure of similarity is the Euclidean distance, which can be computed over many spaces or planes. When comparing two documents, a simple match gives a measure of the Euclidean distance between the documents. "Characteristics" are determined for each of the documents. A list of terms that are either present or not present in the document is the usual means to determine characteristics. The simple match gives the ratio of those characteristics present in both documents plus those absent in both documents to the total number of characteristics. Table 2 lists the terms Losee (1998) used in describing the simple match, computed as shown by Equation 2.4.

Term	**Description**
a	number of characteristics present in both vectors compared
b	number of characteristics present in only first vector compared
c	number of characteristics present in only second vector compared
d	number of characteristics present in neither vector compared
T	sum of $a + b + c + d$

Table 2: Simple Match Terms used by Losee

Equation 2.4 Simple Match $= \dfrac{(a+b)(a+c)+(d+b)(d+c)}{T^2}$

A variation on that method that has been shown to be useful is the Jacquard coefficient. Jacquard's similarity measure eliminated terms that appear in neither document from the overall calculation. In very simple terms, Jacquard's measure can be expressed as the intersection of the sets of characteristics divided by their union. Another frequently used similarity measure is Dice's coefficient, but it has been shown to yield the same ranking as Jacquard's coefficient (Losee, 1998). Jacquard's coefficient can be expressed as:

Equation 2.5 Jacquard measure of similarity $= \dfrac{a}{a+b+c}$

Another widely used measure of similarity is the vector space or cosine measure. Losee (1998) described the measure that was first proposed by Bhattacharyya in 1946. This approach looked at the characteristics of a document as a vector. The similarity between two documents can be measured as the cosine of the angle between the vectors. Computationally, this is a ratio of the inner product of the two vectors to a normalization factor of the lengths of the vectors (Losee, 1998, Salton & McGill, 1983). If $\vec{X}$ represents vector X with elements x_1, x_2, ..., x_n and $|X|$ is the length of $\vec{X}$ and $\vec{Y}$ represents vector Y with elements y_1, y_2, ..., y_n and $|Y|$ is the length of $\vec{Y}$, and θ represents the angle between the two vectors, the cosine of that angle can be computed as:

Equation 2.6 Cosine $\theta = \dfrac{\sum_{i=1}^{n} x_i \bullet y_i}{|X| \bullet |Y|}$

One of the early automatic document processing systems was the SMART (System for the Manipulation and Retrieval of Text) retrieval system designed at Harvard

University in the 1960's. Many retrieval and similarity algorithms were developed for the SMART system as it evolved. One of these was the vector space analysis approach. In this approach, each document is viewed as an n-dimensional vector with each unique word in the document being a dimension. The magnitude in each dimension is the value of the frequency of that term in the document. Two distinct measures can be used to determine similarity in the vector space model. One approach uses the magnitude of the difference of the normalized vectors, but this will result in short similarity queries being shown at a long distance from long documents. Another more widely accepted approach is to measure the similarity as the cosine of the angle of the vectors representing the document and a query or another document. This could result in the opposite bias, with short documents shown with lower similarity scores (Chakrabarti, 2003).

Latent semantic indexing was an enhancement to the vector space model, which was also applied to the SMART system. Matrix transformations were used to project each document into *r*-dimensional space, where *r* is the rank of the matrix. A system of equations could be formed and the Eigenvectors that form the solution set of those equations could be used to estimate the similarity of the documents. The problem with this approach was that systems of equations that large require immense computational resources to reach solution within an acceptable tolerance. Often, to reduce the number of dimensions, key terms were selected rather than every term in the document. In other words, a subset of the original ranking values was usually used to approximate the original vector space. Latent semantic indexing is advantageous in that it provides a way to incorporate semantics by grouping together vectors with similar terms, but there are also drawbacks. The number of dimensions chosen can have a dramatic effect on the result of this measure with this method (Salton & McGill, 1983). In addition, the computational expense is currently prohibitive for large documents sets such as the World Wide Web (Chakrabarti, 2003).

Haveliwala, Gionis, Klein and Indyk (2002) presented an overview of ways to evaluate the similarity in Web-based documents. These authors showed the theoretical derivation of their algorithm to determine how well a specific search strategy works based on Web hierarchies rather than user-feedback. This was applied to content-based approaches utilizing specific words, link-based approaches that used document identifiers and anchor-based approaches that made use of words in or near a specific link to another document. Distance functions were defined according to the distance between classes in a specific hierarchy and an ordering within a family of documents was calculated based on the compatibility of document pairs. The authors' research indicated a Jacquard coefficient was an adequate measure of document similarity for this evaluation.

A study by Ganesan, Garcia-Molina and Widom (2003) pointed out three ways of looking at the similarity of data objects. If all terms in a document are considered a bag, the intersection of the elements in the bag can provide one measure. This is the approach used in simple match, Jacquard and Dice similarity measures. Another way to evaluate similarity is by treating each object as a vector in n-dimensional space and calculating the cosine of the vectors representing the objects. These authors propose enhancing those models with the knowledge contained in a hierarchy describing the data. The approaches suggested by these authors are analyzed to determine if variations of this approach are useful in the specific case of calculating similarity of ontological annotations. These authors suggest different possible measures for similarity of data when organized in a hierarchical manner.

Ganesan, Garcia-Molina and Widom (2003) described their approach as being intuitive and demonstrated it through a specific example. This approach required that the data be organized into a hierarchical tree. No constraints were placed on the shape of the tree, other than the fact that it fan out from a single root node. The actual data items formed the leaves of the tree, while the interior nodes described the classification of those items. The authors presented a simple example and defined two measures of similarity based on those examples. An abbreviated form of those examples is given here and the use of this methodology was demonstrated with semantically annotated documents in this dissertation.

Ganesan, Garcia-Molina and Widom (2003) presented an example depicting music CDs. From the root node, the music was classified as either rock or classical. At the next level, the rock music was classified as Beatles or Stones, while the classical music was classified as Mozart or Chopin. The leaf nodes were albums of that type music by one of those musicians. This is depicted in Figure 3.

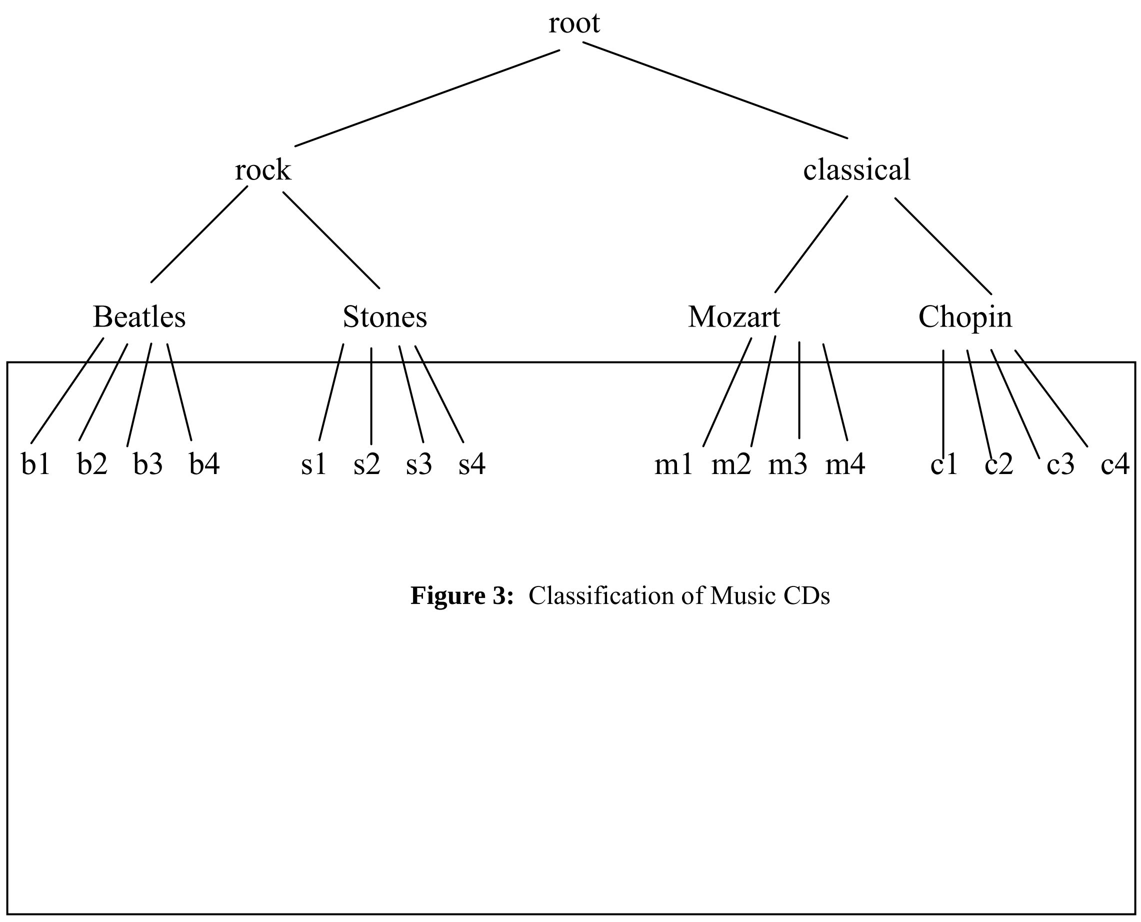


Figure 3: Classification of Music CDs

If the first customer purchased the first two Beatles' albums and the second customer purchased the second two Beatles' albums, traditional similarity measures would not show these two purchases as being similar, since there were no common elements. Ganesan, Garcia-Molina and Widom (2003) proposed two measures that show the similarity of these items. In the first measure, known as the Generalized Cosine-Similarity Measure (GCSM), the similarity of these two purchases would be calculated at 0.8, rather than the zero that would be the result of the intersection-based calculations or the vector-space calculation. The result is arrived at through the following computations.

The dot product of any two leaf nodes was defined as twice the depth of the lowest common ancestor (LCA) divided by the product of depth of the two leaf nodes. The dot product of the two vectors representing the purchases is the summation of the dot products of the leaf nodes. The similarity is then computed based on the traditional cosine measure of the dot product of the two vectors divided by the product of the dot product of each vector multiplied by itself. Ganesan, Garcia-Molina and Widom (2003) used the following formulas to express these computations. (Weighting factors were assumed to be 1 for this example and were omitted to simplify the formula display):

Equation 2.7: $\vec{l_1} \cdot \vec{l_2} = \dfrac{2 * depth(LCA_U(l_1 \cdot l_2))}{depth(l_1) + depth(l_2)}$, where l_1 and l_2 refer to the leaf nodes, and $LCA_U(l_1 \cdot l_2)$ denotes the LCA of leaves l_1 *and* l_2. The subscript $_U$ denotes that this is the point where the two branches containing each leaf unite.

Equation 2.8: $\vec{A} \cdot \vec{B} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j \cdot \vec{l_i} \cdot \vec{l_j}$, where a_i and b_j represent the weighted count (and are 1 in this particular application) and n is the number of leaf nodes

Equation 2.9: $GCSM\ sim\ (A, B) = \dfrac{\vec{A} \cdot \vec{B}}{\sqrt{\vec{A} \cdot \vec{A}}\sqrt{\vec{B} \cdot \vec{B}}}$

According to this definition, $\vec{l_1} \cdot \vec{l_1} = 1$. In the above example, $\vec{l_1} \cdot \vec{l_2} = \dfrac{2 \times 2}{3+3} = \dfrac{2}{3}$

$\vec{A} \cdot \vec{B} = \dfrac{2}{3} + \dfrac{2}{3} + \dfrac{2}{3} + \dfrac{2}{3} = \dfrac{8}{3}$ and $GCSM\ sim\ (A,B) = \dfrac{8/3}{1 + 2/3 + 2/3 + 1} = .8$

The authors, Ganesan, Garcia-Molina and Widom (2003), also defined another similarity measure, known as the Optimized Genealogy Measure (OGM). This measure compares two collections that have a common leaf node and defines a match for any leaf node as being the LCA of that leaf. The leaf similarity is the depth of the match value divided by the depth of the leaf node. The OGM similarity is the sum of all the leaf similarities. If weighting factors were any value other than 1, the OGM similarity would be the weighted average of sum of the products of leaf similarities multiplied by the weighting factors. In this example, since the weighting factors are all 1, this can be expressed through the following formulas:

Equation 2.10: $match_{T1,\ T2} = LCA\ T_1,T_2\ (l_1)$, where T_1 and T_2 denote the tree representation of each collection

The best match is the LCA of an element of T_2 that is also an element of l_1. If there are no common ancestors, the value will be 0 and if l_1 is an element of T_2, the value will be 1. The OGM measure is the weighted average of the individual leaf similarities in the collection.

Equation 2.11: $leafsim_{T1,T2}(l_1) = \frac{depth(LCA_{T1,T2}(l_1))}{depth(l_1)}$

Equation 2.12: $OGM\ sim(C_1,C_2) = \frac{\sum_{l_1 \in C_1} leafsim_{T1,T2}(l_1) * W(l_1)}{\sum_{l_1 \in C_1} W(l_1)}$ where C_1 *and* C_2 denote the collections with a common leaf node

The OGM is an asymmetric measure, meaning that the similarity of C_1, C_2 is not always the same as the similarity of C_2, C_1. According to the authors, an average of the two values can be used to obtain a symmetric measure. In this example, in finding the similarity between A and B, it is necessary to sum the leaf similarity values of l_1 and l_2. Each of these values is 2/3, since the lowest ancestor (Beatles) is at a depth of 2 and the individual nodes are each at a depth of 3. The average of the two measures is then 2/3 or 0.67.

Ganesan, Garcia-Molina and Widom (2003) also defined "second generation" measures of similarity. These measures apply to situations where there are many-to-one relationships (multiple occurrences of the same node), or multiple occurrences of nodes at the same level of the tree. Two measures, known as balanced genealogy measure (BGM) and recursive genealogy measure (RGM) were designed to handle the problem that additional matches at the same level of the hierarchy may not mean that the documents are more similar. The authors pointed out that the situation where the measures will be used would dictate the most appropriate measure. In the case of annotations from an ontology, each annotated term was important and therefore the "first generation" measures were more appropriate. There are no many-to-one relationships, since each term of the ontology is either indicated or not.

Ehrig and Sure (2004) addressed similarity for ontologies in terms of mapping concepts from one ontology to another one. Core to their work was presenting an approach to defining a mapping function so that for each concept in a particular ontology, the function would map to the corresponding concept in another ontology. Various measures of similarity were calculated and evaluated. According to these authors, an ontology consists of: concepts that are arranged in a hierarchy; relations that exist between concepts; instances of the concepts are connected by property instances; and axioms that can be defined to infer knowledge. The authors defined a specific set of rules to use in mapping the concepts. The most basic rules deal with entities within the ontology, denoted as e_{ij}. The entities are elements of the set of concepts, relations and instances for the *i*th concept and *j* represents the entity index within that set. The first rule stated that if two labels are the same, the entities represented by those labels are the same. Higher rules are more dependent on descriptive logics. For example, the fifth rule stated that if the super-concepts from two ontologies are the same, the actual concepts are similar. The researchers proposed that a combination of rules leads to better mapping results than any rule used individually. The combination of rules can be integrated through a weighted similarity function:

Equation 2.13: $sim(e_{i1j1}, e_{i2j2}) = w_k \sum_{k=1}^{n} w_k sim_k(e_{i1j1}, e_{i2j2})$ where w_k is the weight for a specific method sim_k,

Ehrig and Sure (2004) stated that the weights could be assigned manually, or could be discovered by maximizing a training set's F-measure (Yang & Liu, 1999), which quantifies the best number of mappings and is a combination of the precision and recall measures explained below. Ehrig and Sure suggested the use of a neural network to determine how to combine methods for similarity calculation. They also proposed a more sophisticated approach, utilizing a sigmoid function that has the effect of giving more weight to those methods with a high similarity value and a very low weight to those with a low value. Metrics used by these researchers were defined by:

Equation 2.14: Recall $r = \dfrac{\#correct\_found\_mappings}{\#possible\_existing\_mappings}$

Equation 2.15: Precision $p = \dfrac{\#correct\_found\_mappings}{\#all\_found\_mappings}$

Equation 2.16: F-Measure $f = \dfrac{(b^2+1)pr}{b^2 p + r}$ where b is a factor to weight precision and recall and b=1 (the value utilized by these authors) indicates that equal weights are given to precision and recall measures.

The authors found that more advanced combination methods resulted in higher precision and recall and that naïve combinations could sometimes make results worse. While this study concentrated on mappings between ontologies, the general guidelines could be applied to calculations of similarities of annotations from the same ontology.

Another approach is to rank the result of queries involving semantic associations (Anyanwu et al., 2005). These authors have pointed out that relationships in the Semantic Web become more important than individual objects. A method of ranking query results that allowed users to effect the ordering of query results was presented that builds on the idea of the importance of relationships. Many possible ranking schemes were presented. Two terms defined by the authors are *customizability* (the ability for users to select an appropriate ranking scheme) and *flexibility* (the ability for users to apply different ranking schemes to the results). The ranking approach suggested by these authors is based on measuring how much information the user would gain by a particular result. As part of this ranking, a Semantic Match (S-Match) was defined to be the maximum of the semantic match (SemMatch) values. The authors view the instances of RDF property sequences as a labeled path in a knowledge base. The distance between any two properties would be the length of the shortest path connecting those properties (minimum number of nodes that must be visited to travel from one property to another along the path). For a property sequence PS = $p_1, p_2, p_3, \ldots, p_n$ and a set of keywords K = $\{k_1, k_2, k_3, \ldots, k_m\}$, the degree of the match between the between k_i and p_j was defined by Anyanwwu, Maduko and Sheth (2005) to be:

Equation 2.17: SemMatch(k_i, p_j) = $0 < (2^d)^{-1} \leq 1$, where d is the minimum distance between two properties in a property hierarchy.

The maximum of the SemMatch values was used if two keywords matched the same property. The S-Match value for a particular property sequence $ps \in$ PS was calculated by:

Equation 2.18: $\text{S\_Match}(ps) = \sum_{i=1}^{n} \max_{j=1}^{k} \{\text{SemMatch}(p_i, k_j)\}$

The authors presented calculations based on the S-Match that calculate the information content based on a search mode value that varies from 0 to 1 depending on the user's assessment of whether the search is conventional or designed for discovery. This calculation would result in higher rank values assigned to the most unpredictable paths in a discovery mode and lower rank values for unpredictable paths in a conventional mode. The authors developed a prototype system to calculate semantic ranking based on these calculations. That system returned the top-k results utilizing an algorithm that computes the top-k paths for nodes based on the top-k paths for its children. The author's conclusions were based on synthetically generated data, due to the limitations of existing

RDF data collections. This again points out the need for more advanced tools to facilitate accurate annotation of documents.

In addition to similarity, it is also important to have measures for correlation. In this dissertation, calculating the ontology similarity of document pairs was only one step in the process. Another step was to determine if the document text could be used as an indicator of ontological similarity, and vice-versa. In order to do this, the correlation between common term roots, both expected and unexpected, and the similarity measures was calculated. There are a number of ways of calculating correlation. Pearson's correlation coefficient (Rasmussen, 1992) is used to calculate how two variables are related, assuming continuous normally distributed independent random variables. Spearman's coefficient (Rasmussen, 1992) is a common correlation measure to determine how ranked lists are correlated. The square of the Spearman coefficient is a useful estimate of the correlation confidence that can be assumed. In calculating the ranks for the Spearman correlation coefficient, if two values were equal, the average of the corresponding ranks may be used. The field of artificial intelligence has offered more ways to estimate the correlation. Bayesian networks can be configured to determine if one event is the cause (or is caused by) another event (Losee, 1998).

The following definitions are used by Rasmussen (1992) to describe these common correlation formulas:

Equation 2.19: $S_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2$, where $\bar{x}$ is the mean of the x values

Equation 2.20: $S_{yy} = \sum_{i=1}^{n} (y_i - \bar{y})^2$, where $\bar{y}$ is the mean of the y values

Equation 2.21: $S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$

Equation 2.22: *Pearson's correlation coefficient* = $\frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}}$

Equation 2.23: *Spearman's rank coefficient* = $\frac{6\sum_{i=1}^{n}(rank_{xi-}rank_{yi})^2}{n^3 - n}$

As an example, suppose x represents the number of hours each week a person listens to music, and y represents the number of times that person purchases musical recordings. A sample of this type of data is shown in Table 3.

n	***X***	***Y***	***rank$_x$***	***rank$_y$***
1	6	8	4	4
2	4	5	3	3
3	1	3	1	1
4	3	4	2	2
Rank$_x$ is index of x_n value when x-values were sorted in ascending order. Rank$_y$ is index of y_n value when y-values were sorted in ascending order.				

Table 3: Sample Data for Music Listening vs. Purchases

The mean of the x values is 3.5 and the mean of the y-values is 5.0. Equations 2.20-2.22 are used to calculate each value for this data set.

S_{xx} = $(6-3.5)^2 + (4-3.5)^2 + (1-3.5)^2 + (3-3.5)^2 = 6.25 + 0.25 + 6.25 + 0.25 = 13$

S_{yy} = $(8-5)^2 + (5-5)^2 + (3-5)^2 + (4-5)^2 = 9 + 0 + 4 + 1 = 14$

S_{xy} = (6-3.5) (8-5)+ (4-3.5) (5-5) + (1-3.5) + (3-5) + (3-3.5) (4-5)

= 56.25 + 0 + 25 + 0.25 = 81.5

Pearson's correlation coefficient = $\dfrac{S_{xy}}{\sqrt{S_{xx}S_{yy}}} = \dfrac{81.5}{182}$=0.48

Spearman's rank coefficient = 1 - $\dfrac{6\sum_{i=1}^{n}(rank_{xi-}rank_{yi})^2}{n^3 - n}$ = 1 - $\dfrac{6(0+0+0+0)}{64-4}$ = 1

This example illustrates the difference in the two measures of correlation. While the rankings are perfectly correlated, the raw data correlation as measured by Pearson's correlation coefficient does not show a strong correlation.

Chapter 3

Methodology

Approach

This study investigated the correlation of various measures of similarity to the number of common term roots in documents pairs, as well as the number of common unexpected term roots. Documents included in the study were annotated Web pages containing the abstracts of papers submitted to the International Semantic Web Conference (ISWC) during the years 2002-2004. To facilitate the analysis, a software tool known as Semantic Web Analysis of Similarity (SWAS) was developed.

Documents, which were created with Semantic Web annotation tools and posted on the World Wide Web, were arranged into a structure for analysis. This was accomplished by manually extracting the annotations from each document and parsing those annotations into Resource Description Framework (RDF) triples utilizing an online tool. A master spreadsheet was created with one worksheet for each document containing the document text and the RDF triples for that document. Two similar spreadsheets were also created to store the RDF triples of the ontology in both the DARPA Agent Markup Language (DAML) and Web Ontology Language (OWL) versions. Some documents in this study were annotated using each version.. Another document was created to store the parametric values used for each individual SWAS run. These four documents comprise the input for the SWAS system. A more detailed description of these input files is contained in Appendix C.

The SWAS system utilized this input data to determine the relationship of the ontological concepts, identify common terms in the documents and use semantic knowledge provided by the markup of these documents as well as the relationships of the ontological concepts provided through the ontology itself. The root of each term in the document text and the annotations was determined by eliminating common suffixes from each term. Based on whether or not the root of a specific term was present in the relevant annotations for the document, the terms were classified as expected or unexpected. A selection algorithm developed for the SWAS system was used to identify the documents to be included in the analysis. This was done by selecting those documents that do not exceed a user defined threshold for the average difference between the precision (Ehrig & Sure, 2004) and the normalized ranking of common terms in the set of selected documents, according to equations 3.3 - 3.4. An iterative scheme was employed to add those candidate documents to a selection set with the minimum difference between the precision ranking and the ranking of common terms. In the next step, the average difference was calculated for each document in the selection set and those documents whose average difference exceeds the user-defined threshold were deselected. The user was able to define the maximum number of times a document may be deselected from the set before it is eliminated from further consideration, but analysis showed that when the selection set reached a steady state no further changes would happen. The number of deselections was held constant for the data analyzed, allowing all conditions to be reviewed after that steady state had been achieved. User input also defined the minimum number of common terms needed for two documents before that pair would be considered in the selection set. Data was analyzed to show the effect of no common terms versus at least one term in common between selected document pairs. For the selected documents, pair-wise calculations of similarity based on both common terms and common unexpected terms were computed. Six different measures of similarity were calculated in an attempt to define the one most appropriate to measure similarity in this type of data, and the appropriateness of each similarity measure was evaluated through the use of two different correlation calculations. The measures of similarity used were the simple match and Jacquard measures described by Losee (1998), the vector-space or cosine measure (Liu, Ma and Yu, 2001), the Generalized Cosine-Similarity Measure (GCSM) tree-based method proposed by Ganesan, Garcia-Molina and Widom (2003), and two measures proposed in this dissertation that combined both traditional and hierarchical measurement techniques. These new measures are referred to as the unweighted hierarchical measure (UHI) and the weighted hierarchical measure (WHI). Rasmussen (1992) describes various ways of calculating correlation. The correlation techniques used in this dissertation are Pearson's correlation coefficient, which is used to calculate how two variables are related, assuming continuous normally distributed independent random variables, and Spearman's coefficient, which compared the ranked values of these measures for both the common terms and the common unexpected terms. Analysis of the results was done to determine if the additional knowledge found in the form of common terms not expected from the ontological annotation was useful.

Based on the analysis of the results provided by the SWAS system, suggestions for further research were formulated. Recommendations include whether similarity calculations based on all similar terms or the unexpected similar terms were shown to be the more appropriate measure from this study, which measure of similarity for semantically annotated web pages was shown to be the most useful and whether raw values or ranked values were shown to be more appropriate for use. An attempt was made to quantify how effectively similar annotations describe different documents. A proposal was formulated for the development of a tool to be used on web portals to allow novice users to compare semantic annotations to a standard body of annotated documents developed and in use by the community of users utilizing that portal.

Prior to analysis done by the SWAS tool, data was extracted from the documents, including both the semantic annotation and the text. A parsing tool was used to convert the ontology itself as well as the document annotations to RDF triples. The resulting triples, as well as the document text, were stored in spreadsheets. Three spreadsheets were produced. One spreadsheet contained the triples generated from the annotations and the other two contained the triples generated from the ontology itself, in both DAML+OIL and OWL formats. These spreadsheets formed the input data for the SWAS analysis. A fourth input file contained the user-defined input values for each data run.

Parametric studies were conducted on the data. The first variable analyzed was the rules by which valid annotations were selected. In addition, the effect of considering just the annotated terms or all those terms inferred from the annotation, the number of common documents required, and the threshold used for document selection were also analyzed. The ISWC ontology being used described concepts and properties concerning many things about the documents. The only annotations of interest in this particular study were those that described the text contained in the document text. For example, most of the documents contained annotations concerning the university affiliation of the document authors. While this is useful information in many contexts, it is not of interest in this particular study. Other parameters varied were the thresholds to filter out all but the data most pertinent to this study and the number of common terms needed to consider a document pair in the analysis.

The SWAS analysis tool analyzed the documents and calculated ontotological similarity measures. The text of the documents was examined to discover similar terms. Terms were grouped according to the root word of the term by systematically deleting common suffixes from the term. This allowed terms that are grammatically similar (such as the singular and plural versions of the same noun) to be considered as parts of the same root word. The inclusion of all terms having the same suffix was utilized by Fox (1990) in development of a stop list for general text. Further refinements, such as more complex methods of ways to determine the term roots and grouping terms that are synonyms, were left for future exploration and suggested as a needed area for future research. Results of the similarity analysis were evaluated for correlation. Based on results of that correlation, suggestions were made for a Web portal tool to assist in evaluation of the appropriateness of annotations of Semantic Web Documents (SWDs).

In the algorithm used in the SWAS tool, which is a modification of that designed by Liu, Ma and Yu (2001), a set of concepts was generated from the semantic annotations, rather than using a user page. The approach was to draw from a collection of documents annotated according to the same ontology, allowing a comparison of similar terms and concepts in the metadata. “Concepts” in this ontological sense took on a slightly different meaning than that used by Liu, Ma and Yu (2001). Those authors defined concepts as terms co-occurring within the same sentence. In this ontological meaning, concepts were defined by the relationships between terms, or statements in the RDF vocabulary. In the semistructured person example previously presented, the person “Alan” had an attribute of “age” which is 42. In a more complete definition of any person, many attributes would be specified and linked through the relationships. Alan may be married to Jane and have children Cindy and Bill. The relationship between Alan and Jane is different from that between Alan and Cindy and this information would be specified through the ontology. Ontological markups based on the evolving standards proposed by W3C provided a structured way that this knowledge could be embedded into a Web page about a person, so that the entire concept could be captured. In defining ontological similarity, the weighting scheme described by Liu, Ma and Yu (2001) was not necessary, since each attribute is either present or not present in the ontology. The scheme was expanded to allow consideration of the relationship between attributes, as defined in the ontology. Further refinements were made to allow analysis of not only concepts that are the same in different documents, but also concepts that have a similarity based on the hierarchical structure of the ontology.

A matrix structure was utilized to represent the documents and associated term roots. Each document was filtered to remove common non-descriptive terms (such as *a, the, one).* The remaining terms in each document were translated into their root word through suffix elimination. For example, the terms *ontology, ontologies* and *ontological* were considered forms of the root term *ontolog.* The terms were organized into a matrix, where each term formed a single row and each document formed the columns, with an attribute specified for each term of each document. This attribute was a flag denoting if this term is unexpected from the ontology. Those terms that appeared in the annotation concepts or are subclasses in the ontology of the annotated concepts were considered expected knowledge and other terms were designated as unexpected. This matrix was called the *term document matrix*. Similarity measures of the ontology were compared to the terms found common in each appropriate document pair, as well as to those common terms that are not expected from the ontological information.

Data

The primary data set\ for this dissertation was the annotated abstracts of accepted papers for the first three years of the International Semantic Web Conference (ISWC), held in 2002-2004. Those submitting papers to be considered for acceptance were asked to annotate the abstracts in the first year and some authors continued to annotate the documents for the second and third years. Two RDF ontologies were published for use in the annotations. The documents in the first two years were annotated according to the DAML+OIL version of the ontology and in the third year, the documents were annotated according to the OWL version. It was assumed that this group of authors had more knowledge of semantic annotation than the average web user since they were all writing about the Semantic Web. The authors were provided links to several annotation tools, as well as detailed screen shots depicting use of these tools. The authors were encouraged but not required to utilize the provided ontology. The links to the online publication of these documents as of the time of this writing are given in Table 4. The primary document set from the each year of the conference can be found through a search engine using keywords “International Semantic Web Conference annotated papers.”

Document	URL
ISWC Ontology DAML Version	http://annotation.semanticweb.org/iswc/iswc.daml
ISWC Ontology OWL Version	http://annotation.semanticweb.org/iswc/iswc.owl
2002 Annotated Documents	http://annotation.semanticweb.org/iswc/documents.html
2003 Annotated Documents	http://annotation.semanticweb.org/iswc2003/
2004 Annotated Documents	http://annotation.semanticweb.org/iswc2004/annotated_docs/

Table 4: Online Resources for Ontologies and Annotated Documents

Specific concepts in the ISWC ontology make it appropriate for this type of text analysis. Concepts in this particular ontology were designed to describe the content of the paper being summarized in the abstract. Specific objects were designed into the ontology to provide a base of choices concerning the content from which the annotators may choose. In the ontology, the objects were identified by both a universal resource identifier (URI) and a string description of that URI. The string description was provided through the literal data type in the Resource Description Framework (RDF).

The data set for this study was the set of those published annotated abstracts whose annotations conform to the specified ontology, were syntactically correct enough to be processed by an RDF-parser and specified information about the topic of the paper. Each document was a Hypertext Markup Language (HTML) file marked up semantically. The semantic annotations contained references to other Web resources including the ontology itself and at least one resource document that contained the URIs for that Web page. In most cases, the URI of the web page was housed on the server of the institution where the research took place. This resource provided the semantic information about the documents content, the authors of the document, the research and funding that enabled the project to which the documents refers and other information not specific to the content of the document.

The process used in this dissertation to analyze ontological similarity involved, among other things, selection of the most appropriate documents from the data set, determination of the similarity of the ontological annotations, grouping of terms to the term root, identification of unexpected terms, and the correlation calculations between the ontological similarity and the number of common terms and common unexpected terms.

Initially, the set of documents and terms to be considered from those documents had to be identified and organized into some structure. In order to do this, both the ontology and the documents were parsed and the relationships deemed relevant were identified. The semantic annotation code in the documents was parsed using World Wide Web Consortium (W3C) Online RDF Validation Service online tool at http://www.w3.org/RDF/Validator/.

Documents and the ontologies are retrieved and stored prior to the analysis. The ontologies and the annotations in the documents were parsed using the W3C tool. This determined the subject, predicate and object of each triple. The resulting RDF triples were stored in spreadsheets for further processing, which became one of the input data files for the SWAS tool. Similarity comparisons were based only on those items whose subject is the paper itself. This is an important distinction. A triple describing the topic but having the subject as the author of the paper would mean that this was a topic of interest for that author, but not necessarily the topic of this particular paper. Only those classes in the annotation describing the content of the paper and containing specific values for the object of that description were used. Rules were applied by the SWAS tool to ignore those triples that are not of interest in this study. The ontology specified three classes where the subject could be the title of the paper and also refer to the content of the paper. These are “topic”, “formal language” and “tools”. Specific values of the object of the concept were specified in the ontology for the “topic” and “formal language” concepts, but not for the “tools”. Since the ontology did not specify a list of possible tools, only the topic and formal language classes were used. Separate analysis was done on the document set meeting each of these rules, and is described below:

Rule 1: (Subject = title of paper $\wedge$ Predicate = formal language)

Rule 2: (Subject = title of paper $\wedge$ Predicate = topic)

Rule 3: (Rule 1 $\vee$ Rule 2)

The ontology provided a choice of thirty-seven topics and ten formal languages, all of which could be used as the object of the paper itself in an RDF statement. The author also was given the option of defining a specific object not delineated in the ontology. This can be done in both the OWL and DAML+OIL formats. Since those author-supplied objects were not consistent across documents and are used very infrequently in this document set, this data was not included in the analysis. Only those concepts utilizing objects specified in the ontology were considered. In some cases, the topic was a subclass of another topic. That information can be applied in two different ways. The superclass topics could be considered as inferred knowledge in the similarity analysis and those same terms could be flagged as expected terms in the knowledge discovery phase.

Selection Algorithm

The ultimate goal of this research project was to propose a method to provide validation of the annotation in candidate SWDs as compared to a set of documents deemed appropriate for this use based on a specific ontology and accessed through a Web portal. It is fully acknowledged that there are many goals of semantic annotation and not every document is appropriate for every application. That in no way implies that the document is not appropriately annotated for its purpose, but does mean that not every document was appropriate for this particular application. Prior to performing a pair-wise analysis of common terms in the document set and correlating that data with the similarity measures between document pairs, the best documents for this specific application were selected. Ganesan, Garcia-Molina and Widom (2002) suggest that a Top K method be employed; that is, the top k number of matches were the only ones considered in the analysis. The problem with applying that technique to this study is that the primary focus of the study being undertaken is a comparison of various measures of similarity and no one of these measures could be used to select the best matches. Threshold values were varied to include all documents, and approximately each quartile of the possible documents, as well as a smaller set of documents including the best document set with ten to thirty documents. In all cases, no data was collected for sets that contained less than ten documents.

Ehrig and Sure (2004) used the term *precision* to mean the ratio of correct mappings to all found mappings. This same idea was used to measure the precision of the various similarity measures. In some way, all measures considered in this study depended either on the ratio of the number of annotated concepts present in both documents in a document pair to the number present in either document or the ratio of the number of lowest parents of those concepts present in both documents to the number of lowest parents in either document. The following equations show how these values were calculated when using documents *a* and *b* from the set of documents.

Equation 3.1: $$docRatio_{ab} = \frac{number\_annotations\_in\_both}{number\_annotations\_in\_either}$$

Equation 3.2: $$parentRatio_{ab} = \frac{number\_annotations\_sameparent\_in\_both}{number\_annotations\_sameparent\_in\_either}$$

One aim of this study was to determine if there is a correlation between the similarity measures and the common terms between document pairs. This was accomplished by ranking the common terms and the similarity measures, normalizing these measures so they had a value between 0 and 1, and calculating a difference function for each similarity measure to determine, for each document pair, the difference between the ranking of similarity measures and that of the common terms or common unexpected terms. Selecting those document pairs with the minimum difference produced the subset of documents for this analysis. By using the *docRatio* and *parentRatio* measures rather than any specific measure of similarity, a candidate set of documents was designated for use in the measurement comparisons. In order to gain an overall measurement of the appropriateness for this analysis, an average difference for each document as compared to the others used was calculated. As new documents were added to the candidate set, the two documents with the lowest difference may have had an extremely high difference when compared to any other document in the data set, changing the average difference measure for each document. For this reason, an iterative scheme was employed to reevaluate the appropriateness as each new document is considered for the final set. It was determined that all documents sets reached a state where no further improvement could be made if the number of deselections was set to five. One of the parameters varied in these studies was the use of the docRatio or parentRatio to calculate appropriateness. The term *ratio* applied to either of those calculations. Another parameter was whether the measures are compared against any common term, or against common unexpected terms. The term *in_common* referred to the ranked value of either of these values. SWAS evaluated cases separately to examine the effect of using common term roots or common unexpected term roots. Separate data was collected allowing documents pairs with no common terms and document pairs that were required to have at least one common term.

Table 5 lists the definitions that were used to calculate the differences. Figure 4 shows the major steps of the algorithm. Equations 3.3–3.5 (contained in Figure 4) illustrate the calculations involved in the difference computations. Initially, *S* was an empty set. The selection scheme then added any document pairs that had the minimum value of $Diff_{ab}$. These documents were added to set *S*. The $ASelDiff_a$ value was then calculated for all documents in the current candidate set. Any document whose average difference exceeded a specified threshold was deselected from the set. A new minimum value of $Diff_{ab}$ was then calculated for those documents not in the candidate set. The process continued until all documents are in set *S* or have been deselected *nds* times. The maximum number of times any document may be deselected was defined by *nds*. Similarity correlations were calculated for those documents in set *S*.

Using the definitions in Table 5, the computational time complexity of the major loop in the algorithm is O(*nd* * *nds*), or the product of the number of valid documents times the maximum number of times each document may be deselected. Within that loop, the complexity of the *m* caluculation in line 4 and the $Diff_{ab}$ calculation in line 8 are O(nd^2). The complexity of the $ASelDif_a$ calculation shown in line 11 is O(*ns*) which is bounded by O(*nd*).

Term	Definition
D	set of valid documents (those which have at least one annotation of interest)
nd	number of valid documents in D
S	set of documents that have been determined to be possible appropriate candidates (those documents currently selected)
ns	number of documents in S (number of selected documents)
th	threshold to denote the maximum value of *Adiffa*
nds	maximum number of times any document can be deselected from D
$nRatio_{ab}$	normalized *ratio* (based on Equation 3.1 or 3.2, depending on case being evaluated)
nIn_common_{ab}	normalized *in_common* between documents a and b

Table 5: Definitions of Terms used for Difference Computations

Equation 3.3 $Diff_{ab} = | nRatio_{ab} - nIn\_common_{ab} |$

Equation 3.4 $$ADiff_a = \frac{\sum_{i=1}^{j} Diff_{ai}}{nd}$$

Equation 3.5 $$ASelDif_a = \frac{\sum_{i=1}^{j} Diff_{ai}, i \in S}{ns}$$

Initialization: Using equations 3.3, calculate $Diff_{ab}$ for all document pairs *ab*.

Using equation 3.4, calculate $ADiff_a$ for each document *a*.

Initialize deselection counter to 0 for each document *a*.

Process:

```
1.  while there are more documents that can possibly be selected:
2.      for all documents $a \in D$, where $a \notin S$
3.          for all documents $b \in D$, where $b \notin S$
4.              calculate $m$ = minimum $Diff_{ab}$ (Equation 3.3)
5.      for all documents $a \in D$, where $a \notin S$
6.          for all documents $b \in D$, where $b \notin S$
7.              if document $a$ has been deselected < $nds$ times
8.                  if $Diff_{ab} = m$
9.                      add document $a$ into $S$
10.     for all documents $a \in S$
11.         calculate $ASelDif_a$ (Equation 3.5)
12.     for all documents $a \in S$
13.         if $ASelDif_a > th$
14.             de-select document $a$ (remove from $S$)
15.             increment deselection counter for document $a$
```

Figure 4: Steps of the Selection Algorithm

A brief example is given below for four documents identified as A, B, C and D using a threshold value of 0.45, a value of 3 for *nds*, no minimum number of common terms and the data in table 6, which reflects the ranked data for the documents. This example is expanded later in this chapter to show calculations of similarity measures as well.

Based on the data shown in Table 6, documents A and C will be added to the selection set, and $ADiff_A$ and $ADiff_C$ will each have a value of 0, which does not exceed the threshold. Now the minimum difference value of the unselected document pairs is 0.25, which will result in documents B and D being added to the selection set. The *ASelDiff* values are now calculated as:

$$ASelDiff_A = \frac{0 + .25 + 0 + .5}{4} = 0.19$$

$$ASelDiff_B = \frac{.25 + 0 + .25 + 1}{4} = 0.38$$

$$ASelDiff_C = \frac{0 + .25 + 0 + .25}{4} = 0.13$$

$$ASelDiff_D = \frac{.5 + 1 + 0 + .5}{4} = 0.5$$

Document D was deselected, since its average difference exceeds the threshold. Since there is a value of 3 for the number of times a document may be deselected, and there is document D is the only document not currently in the selection set, this document was added back in and then eliminated from the selection set two more times. It was determined that no changes in the document set occurred with *nds* values higher than five, so this value was used for the analysis.

Doc Pair	$nRatio_{ab}$	nIn_Common_{ab}	$Diff_{ab}$
A, B	0.75	1.0	0.25
A, C	0.5	0.5	0.0
A, D	0.75	0.25	0.5
B, C	1.0	0.75	0.25
B, D	1.0	0.0	1.0
C, D	0.25	0.0	0.25

Table 6: Sample Data for Selection Example

Similarity Measures

The annotations are analyzed and compared for similarity using several different methodologies. Calculations are done using the following techniques: simple match similarity, Jacquard coefficient, cosine or vector-space analysis based on the research of Liu, Ma and Yu (2001) and the GCSM tree-based method proposed by Ganesan, Garcia-Molina and Widom (2003). It should be noted that while Dice's coefficient is widely accepted as a measure of document similarity, it has been shown to produce the same ranking as the Jacquard coefficient (Losee, 1998) and therefore was omitted from this analysis. In addition to those listed above, two new measures will be evaluated evolving from the work of Liu , Ma and Yu (2001) and Ehrig and Sure (2004).

The calculations involved in the vector-space analysis proposed by Liu , Ma and Yu (2001) were dependent on the cosine measure algorithm based on the vectors representing the two documents being compared. The vectors contained a value for each annotation concept as it relates to the document, with a value of zero denoting that the concept is not present. It should be noted that there are many variations of the vector-space model, especially when considering term weights as a part of the analysis. The basic unweighted measures were used for this analysis and the effect of weighting was evaluated in a measure based on the parent node of the concept in the ontology. Ganesan, Garcia-Molina and Widom (2003) showed how the hierarchical knowledge could be applied to produce a measure of similarity. Ehrig and Sure (2004) investigated the use of combining similarity measures and showed that this process can be more useful than the use of naïve measures.

The first new similarity measure analyzed in this study built on the results of that research. This measure, referred to as the unweighted hierarchical measure (UHI), combined terms according to the parent node and used the vector showing annotations relating to the parent node to calculate the similarity measure utilizing the vector-space analysis employed by Liu, Ma and Yu (2001). The second measure was similar, but was weighted by the frequency measure (based on the number of annotations that relate back to the same parent node) and was called the weighted hierarchical measure (WHI). For the weighted measure, the weighting factor was the percentage of concepts with the same parent node as compared to the total number of lowest parents of unique concepts in each document. Equations 3.6 – 3.8 define the computations for these measures.

Equation 3.6 $$lw_{a,i} = \frac{lp_{a,i}}{nc_a}$$

Equation 3.7 $$UHIsim(d_a\,,\,d_b) = \frac{\sum_{i=1}^{lp} l_{a,i} \times l_{b,i}}{\sqrt{\sum_{i=1}^{lp} l_{a,i}^2} \times \sqrt{\sum_{i=1}^{lp} l_{b,i}^2}}$$

Equation 3.8 $$WHIsim(d_a\,,\,d_b) = \frac{\sum_{i=1}^{lp} lw_{a,i} \times lw_{b,i}}{\sqrt{\sum_{i=1}^{lp} lw_{a,i}^2} \times \sqrt{\sum_{i=1}^{lp} lw_{b,i}^2}}$$

These new measures related to the similarity of the lowest parent in the hierarchical representation of the ontology, or that parent node closest to the root node. In the case of the class in the ontology known as "formal language", the lowest parent was the same as the concept of interest, since no subclasses were defined for the formal languages. The ontology provided for a property "has subtopic" that allowed subclasses of the "topic" class. In the case of topics, there did exist more than one lowest parent for some concepts, since there were no cardinality rules providing that a concept have a single superclass. The lowest parent was designed as a list of all parent nodes of a particular concept of interest representing the topic classes that have no superclasses.

The simple match similarity measure considered all concepts of interest, while the Jacquard measure eliminated those where there is a joint absence in which neither document references that concept. For the purpose of this study, calculations for the cosine measure were adapted to eliminate any weighting based on frequency of occurrence of terms, since a term is either indicated in the ontology or not. The definition of GCSM developed by Ganesan, Garcia-Molina and Widom (2003) was expressed using the same terminology as the other measures to more clearly illustrate the relationships between measurements. An expanded term-document matrix was used for calculations, showing not only which terms were actually annotated for the documents, but also which terms could be inferred from the hierarchical knowledge about the annotated terms provided through the ontology. Measures were calculated utilizing ontological annotations only as well as the inferred annotations. Table 7 lists the formal definitions of terminology for the new measures of similarity between documents a and b adapted from those specified by Liu, Ma and Yu (2001). Formulas for the simple match, Jacquard measure, cosine measure, and GCSM based on these definitions are given in Equations 2.4, 2.5, 2.6 and 2.9.:

Figure 5 illustrates a sample ontology, and Table 8 illustrates sample document annotations using this ontology. In Table 8, a value of 1 meant that document contains an explicit annotation to the concept. In this example, lp = 3, since concepts 1, 2, and 3 did not have a parent concept.

Term	**Definition**
n_c	number of concepts of interest in the ontology
nc_a	number of concepts annotated in document *a*
lp	number of lowest parents of concepts of interest in the ontology
c_i	*i*th concept
la_p	number of lowest parents of concepts of interest in document a
$lp_{a,I}$	number of concepts in document a which have lowest parent l_i
l_i	lowest parent of c_i
l_{ai}	lowest parent indicator, value of 1 indicates some concept is present which has lowest parent *i* in document *a*, 0 indicates no concept is present in document *a* which has lowest parent *i*
$lw_{a,I}$	weighted value of $l_{a,I}$
$UHIsim_{a,b}$	similarity of documents *a* and *b* based on unweighted hierarchical measure
$WHIsim(d_a, d_b)$	similarity of documents *a* and *b* based on weighted hierarchical measure

Table 7: Definitions of Terms used in Similarity Calculations

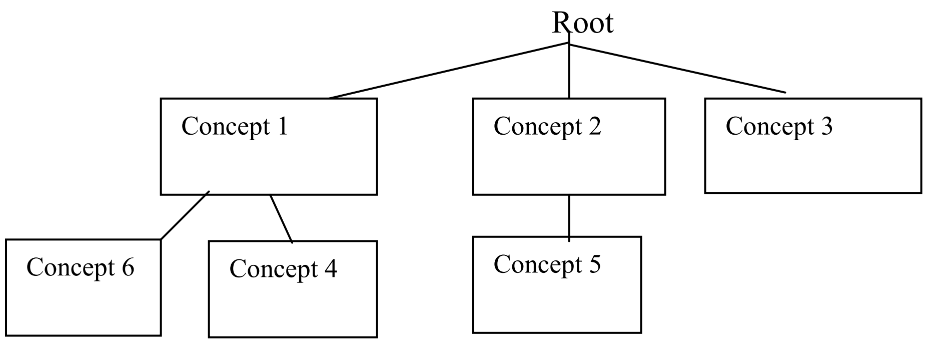


Figure 5: Sample Ontology

	Lowest Parent	**Doc. A**	**Doc. B**	**Doc. C**
Concept 1	1	0	0	1
Concept 2	2	1	1	0
Concept 3	3	1	1	1
Concept 4	1	0	0	1
Concept 5	2	0	0	1
Concept 6	1	1	0	0

Table 8: Document Matrix for Sample Documents with Annotations

By using Equation 3.6, the following weighted values of the parent indicators for the lowest parent concepts could be calculated for each document.

$lw_{a,1}$ = 1/3, since there is 1 annotated concept (concept 6), which has concept 1 as lowest parent, and there are 3 concepts annotated in the document.

$lw_{a,2}$ = 1/3, since there is 1 annotated concept (concept 2), which has concept 2 as lowest parent, and there are 3 lowest parent concepts annotated in the document.

$lw_{a,3}$ = 1/3, since there is 1 annotated concept (concept 3), which has concept 3 as lowest parent, and there are 3 concepts annotated in the document.

$lw_{b,1}$ = 0/2 , since there is there are no annotated concepts which have concept 1 as lowest parent, and there are 2 concepts annotated in the document.

$lw_{b,2}$ = 1/2, since there is 1 annotated concept (concept 2), which has concept 2 as lowest parent, and there are 2 concepts annotated in the document.

$lw_{b,3}$ = 1/2, since there is 1 annotated concept (concept 3), which has concept 3 as lowest parent, and there are 2 concepts annotated in the document.

$lw_{c,1}$ = 2/5, since there are 2 annotated concept (concept 6), which has concept 1 as lowest parent, and there are 5 concepts annotated in the document.

$lw_{c,2}$ = 1/5, since there is one annotated concept (concept 5), which has concept 2 as lowest parent, and there are 5 concepts annotated in the document.

$lw_{c,3}$ = 1/5, since there is one annotated concept (concept 3), which has concept 3 as lowest parent, and there are 5 concepts annotated in the document.

Equation 3.7 can be used to find the unweighted hierarchical similarity between documents, and equation 3.8 can be used to find the weighted hierarchical similarity. To find this value for document B and document C, the following calculations would be used:

$$UHIsim(d_b\,,d_c) = \frac{(0\times 1)+(1\times 1)+(1\times 1)}{\sqrt{(0)^2+(1)^2+(1)^2}\times\sqrt{(1)^2+(1)^2+(1)^2}} = \frac{2}{\sqrt{2}\times\sqrt{3}} = 0.82$$

$$WHIsim(d_b\,,d_c) = \frac{(0\times 2/5)+(1/2\times 1/5)+(1/2\times 1/5)}{\sqrt{(0)^2+(1/2)^2+(1/2)^2}\times\sqrt{(2/5)^2+(1/5)^2+(1/5)^2}}$$

$$= \frac{2/10}{\sqrt{1/2} \times \sqrt{6/25}} = 0.58$$

When evaluating the similarity between documents A and C, the unweighted value was 1. This indicated that there was an annotation with each possible lowest parent in each document. The weighted value of 0.94 indicated that there is some difference in the number of annotations mapping back to each parent.

Overview of SWAS

The SWAS tool was developed in Java, with connections to the input spreadsheets provided via a Java database connectivity / open database connectivity (JDBC/ODBC) bridge. The ODBC connection to the file must be established in the operating system prior to using the SWAS tool. Information concerning the setup of the connection for a Microsoft XP operating system was organized, and is shown in Appendix E. Prior to processing by the SWAS system, the ontology and document data was extracted and RDF statements were parsed through the use of an online tool, as described in the “Data” section of this chapter. Two single worksheet spreadsheets were used to store the ontology data in DAML+OIL and OWL formats. A multi-worksheet file stored the data for each document as a separate worksheet in the spreadsheet, and a text file contained the input parameters. Full explanations of the input is available in Appendix C.

The SWAS tool was designed to produce data concerning the correlation of common terms and common unexpected terms to that of the ontological similarity as measured in various ways. The source code is provided in Appendix F, and is available online at http://home.cfl.rr.com/lookhome/joelynn/SWASsource. A java archive file (JAR) is at http://home.cfl.rr.com/lookhome/joelynn/SWASjar. Parameters varied included the rules to be applied (examining topic concept, formal language concept, or both), document selection method (based on annotated concepts, or lowest parents of those concepts), the values for the threshold for appropriate documents, and the value for the number of common terms needed for a document pair to be considered for selection. Other data such as the specification of the file names for the ontological and document text data and the maximum sizes for arrays was also included. Figure 6 gives an overview of the SWAS tool.

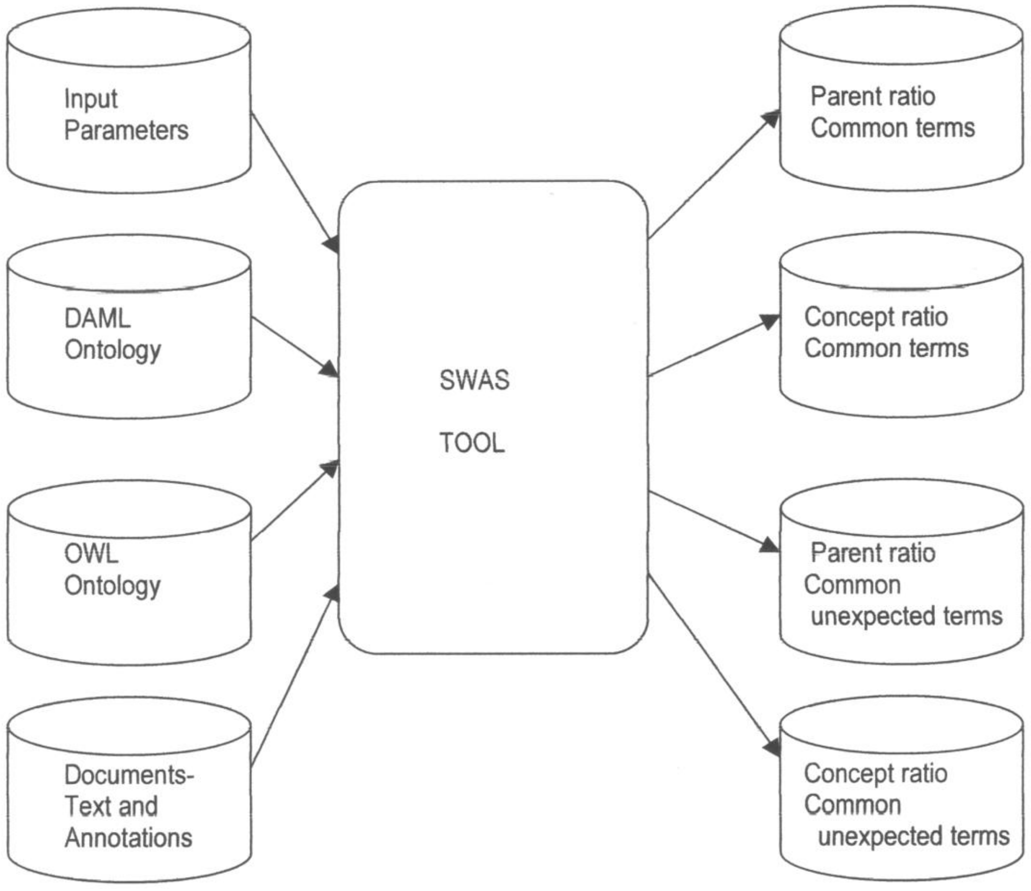

Input Parameters
DAML Ontology
OWL Ontology
Documents- Text and Annotations
SWAS TOOL
Parent ratio Common terms
Concept ratio Common terms
Parent ratio Common unexpected terms
Concept ratio Common unexpected terms

Figure 6: Overview of SWAS tool

Input data was processed by SWAS and four output files were produced for each parametric run, which were imported into a spreadsheet to allow further analysis as input parameters were varied. The data was organized into a summary spreadsheet. Highlights of the findings are found in Appendix D. For each parametric run an output file was produced to show the results if document selection was based on the ratio value as calculated by equation 3.1 or equation 3.2; that is whether it was based on the ontological concept, or the parent node of that concept in the ontology. In addition, for each run, the similarity measures and correlations were calculated for both common term roots and common unexpected term roots in the text of each document.

SWAS was developed as a system of Java classes. The main class, called SWAS, served as the focal point from which all other processing was initiated. Several small classes modeled portions of the system and performed specific tasks. The InputParameters class included the module that reads the input data from a text file. The RDFConceptArray class modeled the concept data, providing the arrays to store this data. The class ConceptData modeled a concept derived from the ontology. The class DocText modeled the text found in a document. Both ConceptData and DocText include data items to store the full text and the root words of that text for the concepts and document text. The RDFStmts class modeled each RDF triple in the ontology. The Root class contained a method to determine the root word of a string through elimination of specified suffixes. The Stop class removed terms in the stop list from the concept word list or the document text. The Words class modeled a collection of words, and contained modules to strip special characters from the words and sort the words.

The TermDocumentMatrix class modeled the grid that indicates term roots in each document, contained methods to create and add data to the master grid for each document, create and update a list of root terms for all documents and assign a status of to each term. The status flag showed if the term is present in that document and if the term is also part of the annotation, or if it can be expected from the annotation since it is a part of a parent of an annotated concept.

The Ontology class modeled the ontology, and has modules to read each version of the ontology using a structured query language (SQL) query to obtain the data from the spreadsheet where it was stored after pre-processing. There was also a module in this class to cull the RDF statements to only those of interest, according to the parameters specified. Each RDF statement was examined, and those that have subconcepts were indicated so the lowest parent of each concept can be identified. Another module assigned a level number to each concept. The terms contained in the object of each concept of interest are reduced to root words and stored in a list for that concept. Each term is flagged as either directly annotated, inferred through the hierarchy or not annotated. The Ontology class also contained modules to find the name of a concept given the URI or the concept number and to find the literal identifier for a concept, as specified in the ontology. There was also a method to calculate the similarity of annotated terms using the data calculated in the Ancestors class.

The Ancestors class modeled the hierarchy for the ontology and relates concepts to parents and lowest parent. There were methods to find common ancestors of two concepts, find lowest common ancestors, and to find the dot product vector of leaf nodes for the Generalized Cosine-Similarity Measure (GCSM) measure suggested by Ganesan, Garcia-Molina and Widom (2003) as shown in equation 2.9.

The Annotations class contained the data representing the RDF triples found in each document as well as matrices denoting which of the possible concepts contain annotations in each document, and the parent concepts indicated through the ontology. The instance of the Annotations class received an object of the Ontology class and an object of the InputParameters class as input. The Annotations class contained methods that read a spreadsheet containing the text and annotations from the documents with each document having been stored as a separate worksheet in the spreadsheet. Data was extracted from the file through a SQL query. The data in both the ontology and document spreadsheets had been prepared via an offline process.

The Stats class modeled the statistics for the system. This class contained modules to: select the best documents to use according to the calculations in equations 2.4-2.6; calculate the different similarity measures as indicated in equations 2.4-2.6, 2.9, 3.7 and 3.8; correlate the ontological and textual similarity according to equations 2.22 and 2.23; and write the output files. Separate processing was done for four cases to produce the four output files. Figure 6 outlines the major steps accomplished by the SWAS system. In this diagram, the class name precedes the module name, and these are separated by a dot.

SWAS read the input parameters which specified which option was used to choose the concepts of interest, the thresholds and number of required common terms to be used in the selection algorithm for each output case, the URI and various maximum limits used to size arrays in the data structures. SWAS then read and stored the parsed ontology data and determined superclass lists for each appropriate concept in the ontology, based on the rules specified, as well as the level of that concept in the overall hierarchy.

The next step was to read and store the data concerning each document. The RDF triples describing the ontological annotation as well as the document text were examined, and those document triples meeting the rules specified were stored. SWAS examined the text of the documents, eliminating common non-descriptive terms through a stop list based on the list generated by Fox (1990). Minor modifications were made to that stop list to include common contractions of the terms and to omit those terms that were meaningful to this study. For example, the term “order” was included in Fox’s original stop list, but that particular term could be meaningful in this set of documents, so it was not included on the stop list. Each term was reduced to a root based on suffix elimination. A term document matrix was developed to show which terms in the document text were also in the annotations for that document and which can be inferred from those annotations by use of the superclass list for each annotated concept.

SWAS then selected the appropriate document subset, by successively adding the document pair with the lowest difference between the ranking of the ontological similarity and the number of common terms. All documents in the candidate set were then evaluated to find the average difference for each document. Documents with an average difference higher than the threshold were deselected from the candidate set. This process continued until all documents not included in the candidate set had been deselected a proscribed number of times. In this way, the selection algorithm was used to identify the subset of documents for the analysis. The ontological similarity of that document subset was calculated, as was the correlation between similar documents and the number of common terms for each document pair in the subset. Correlation between similarity measures and common terms was calculated, and a data file containing the results was produced. This file included the subset of selected documents, all of the similarity measures for those documents, the number of common terms and common unexpected terms for each pair of documents and the correlation of these measures. Results were stored in a tab-delimited text file that was imported into standard spreadsheet software applications. Major steps of the SWAS system are shown in Figure 7.

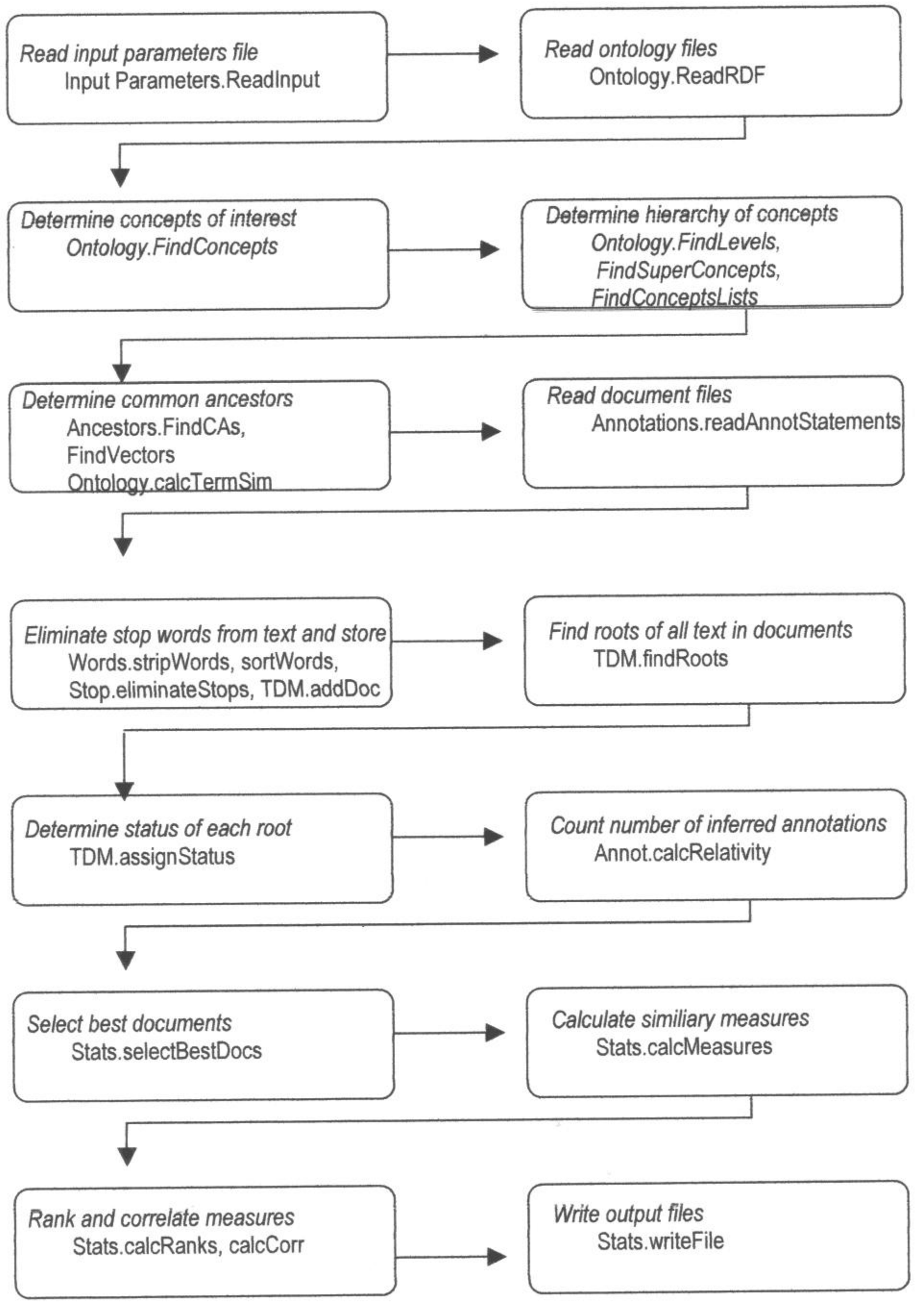

Read input parameters file
Input Parameters.ReadInput
Read ontology files
Ontology.ReadRDF
Determine concepts of interest
Ontology.FindConcepts
Determine hierarchy of concepts
Ontology.FindLevels,
FindSuperConcepts,
FindConceptsLists
Determine common ancestors
Ancestors.FindCAs,
FindVectors
Ontology.calcTermSim
Read document files
Annotations.readAnnotStatements
Eliminate stop words from text and store
Words.stripWords, sortWords,
Stop.eliminateStops, TDM.addDoc
Find roots of all text in documents
TDM.findRoots
Determine status of each root
TDM.assignStatus
Count number of inferred annotations
Annot.calcRelativity
Select best documents
Stats.selectBestDocs
Calculate similiary measures
Stats.calcMeasures
Rank and correlate measures
Stats.calcRanks, calcCorr
Write output files
Stats.writeFile

Figure 7: Major Steps of SWAS System

Example

In order to provide a simplified example of these measures, a mock version of the ontology containing only eleven concepts was provided. Using the ISWC ontology, four sample Web documents were created for this formal dissertation proposal to illustrate the document selection and similarity analysis. The ontology developed for the annotation of documents presented at the First International Semantic Web Conference and these sample documents annotated from that ontology formed the sample data set used in the example in this dissertation. Although the actual ontology was the basis for the annotations, the sample annotated pages that formed the sample data set are entirely fictitious, developed to illustrate the concepts being described in an overly simplified manner. These sample documents served to exemplify the process that was employed and any resemblance to any actual research paper from any source is unintentional and purely coincidental. These sample documents used only a portion of the possible instances proscribed in the conference ontology, so the relationships between the documents could be easily visualized. In addition to the annotated abstract document, a resource document providing needed URIs was also constructed for each of the documents. While this was not nearly as extensive or semantically rich as the resource documents referred to in the actual annotated abstracts, it did provide a simple foundation for the explanation of the process that was later employed.

The following example demonstrated how the similarity and correlation measures were calculated for the sample documents prepared for this proposal. The example documents are found in Appendix A. In preparing these sample documents for analysis, RDF triples were created by parsing both the ontology itself and the annotations within the documents. The DAML+OIL version of the ontology was used for this example.

The annotation of a published SWD can be viewed by viewing the source code of the document via the "View Source" option available on most current browsers and is shown in the document listing provided in Appendix A of this document and at http://home.cfl.rr.com/lookhome/joelynn/OntologyExamples. The RDF triples formed from the annotations can be found published online at http://home.cfl.rr.com/lookhome/joelynn/OntologyExamples/Triples. The parsed graphs of the annotations are also available online and can be found at http://home.cfl.rr.com/lookhome/joelynn/OntologyExamples/Graphs. Based on the concepts identified through the parsed ontology, the objects of those statements where the subject is the article itself is used in narrowing the selection concepts. Rule 2, which states that the subject must be the paper itself and the object must be topic, was used for this example. These objects formed the set of concepts of interest. Although the ontology contains 47 possible concepts, only 11 are used in this example, for the sake of simplicity. In addition, the inference rule applied to augment the explicit annotation of the document was identified. In this example, the inference rule applied was that if a topic was the object of the *"HasSubTopic"* predicate, then the subject of that relationship (the superclass) was included in the expanded matrix.

For the purpose of identification, the documents were labeled A, B, C and D. Since these were created solely to demonstrate the proposed technique, each of the documents contained some annotation for at least one of the concepts of interest. The SWAS tool labeled any document that does not contain at least one annotation for a concept of interest as invalid, and that document is discounted from further analysis. In some cases, the concept was a subclass of another interesting concept. The fact that this concept could be inferred from the annotated concepts was flagged in the document matrix via the status element, to investigate the effect of inferred knowledge (using the superclass designations) on the document similarity measures. In the subsequent phase, when searching for unexpected terms, any inferred common terminology was included in the list of expected terms. Similarity measures were calculated based on actual annotations and on inferred annotations for those measures where the inference was not a part of the measure itself.

Initial calculations were done to determine similarity based on simple match, Jacquard coefficient, modified vector-space analysis, GCSM, and two new measures. Objects selected from the specified concepts form the initial concepts of interest. The expanded concepts of interest also include those objects that are superclasses of the ones explicitly specified in the ontological markups. An initial document matrix was created, with the documents to be analyzed forming the columns of the matrix and the concepts of interest forming the rows. Each column of the matrix formed a document vector that represents the concepts of interest that were indicated through the ontology as present in that document. The matrix was enhanced to show additional concepts of interest, which could be inferred from the ontology through the specified inference rules. For each document, the concepts explicitly specified through the annotation were indicated with a value of 1, the inferred concepts were indicated with a value of 2, and other terms had a value of 0.

The matrix was created to calculate the ontological similarity of the documents. Similarity measures were calculated and used for this example, to determine which measure best correlates to the common terms and common unexpected term roots (CURs) found in the documents. The techniques for simple match, Jacquard coefficient and the cosine measure found through analysis of the vector-space were applied to calculate three measures of similarity for each document pair, as well as hierarchical measures, including one of those suggested by Ganesan, Garcia-Molina and Widom (2003).

Document A indicated that the topics (concepts of interest) were “Text Mining”, ”Semantic Web Languages”, and “Semantic Annotation.” Documents B and D indicated the topics were ”Semantic Web Languages” and “Semantic Annotation.” Document C indicated that the topics were “Agents”, “Logic”, “Text Mining” and “Semantic Annotation”. “Text Mining” was a subtopic of “Knowledge Discovery”. “Semantic Web Languages” and “Semantic Annotation” were subtopics of “Semantic Web”, which was a subtopic of “World Wide Web”. “Agents” was a subtopic of “Artificial Intelligence”, so each of these terms was included in the expanded matrix. The “Logic” resource is not a subtopic of another topic. Two other topics, “Network Infrastructure” and “E-Business” were not indicated topics in any documents in this demonstration. None of the sample documents was annotated with either of these resource topics.

Table 9 depicts the original document matrix, and shows those concepts and the superparent concept indicated by each annotation, along with the flags denoting the annotation of documents with those concepts. It should be noted that the ontology shows the object of a relationship as an identifier with no blank spaces, but also denotes the literal value of that field. For example, the first index “Text mining” is actually denoted as the object of the topic of the paper as “text_mining”, but is then related to the literal value “text mining”. Only the literal values are shown in this chart and the literal values were used in the text analysis.

	Lowest Parent	**Doc. A**	**Doc. B**	**Doc. C**	**Doc. D**
Concept 1 Text mining	8	1	0	1	0
Concept 2 Semantic Web Languages	9	1	1	0	1
Concept 3 Semantic Annotation	9	1	1	1	1
Concept 4 Agents	8	0	0	1	0
Concept 5 Logic	5	0	0	1	0
Concept 6 Knowledge Discovery	8	2	0	2	0
Concept 7 Semantic Web	9	2	2	0	2
Concept 8 Artificial Intelligence	8	2	0	2	0
Concept 9 World Wide Web	9	2	2	0	2
Concept 10 Network Infrastructure	10	0	0	0	0
Concept 11 E-Business	11	0	0	0	0

Table 9: Document Matrix Including Parent Concepts

Table 10 shows the initial ratio calculations needed for document selection, along with the difference values. Table 11 shows the similarity calculations including the semantic inferences for the selected documents as well as the two measures of correlation. It should be noted that these calculations assumed the inferred knowledge from the ontology. The actual SWAS system calculated measures both utilizing and ignoring that inferred knowledge. Appendix B shows the relationship of the term roots to the documents.

Doc Pair	**Doc Ratio**	**Ratio Rank**	**Common Terms**	**Rank Common Terms**	**Diff**
A,B	4/7	.75	4	1.0	.25
A,C	4/9	.5	2	.5	0
A,D	4/7	.75	1	.25	.5
B,C	1/9	1.0	3	.75	.25
B,D	4/4	1.0	0	0	1.0
C,D	1/9	.25	0	0	.25

Table 10: Difference Values for Document Selection

Document Pair	Ssim	Jsim	Vsim	UHIsim	WHIsim	GCSM
A,B	9/11 = .82	4/6 = .67	4/(7x4) = 4/28 = .14	1/(2x1) =1/2 = .5	(4/7)/ $\sqrt{25/49}$ =.8	0.91
A,C	7/11 = .64	5/9 = .56	5/(6x8) =5/48 = .10	2/(2x3) =2/6 = .33	(8/21)/ $\sqrt{25/49}$ $x\sqrt{1/2}$ =.75	0.63
B,C	5/11 = .45	3/9 = .33	3/(4x8) =3/32 = .09	1/(1x3) =1/3 = .33	(1/6)/ $\sqrt{1/2}$ = .12	0.69

Table 11: Similarity Measures for Matrix Enhanced with Inferred Data

Parametric values used for this example were:

Rule Selection: Rule 2 (Subject = title of paper $\wedge$ Predicate = topic)

Ratio: docRatio $\dfrac{number\_annotations\_in\_both}{number\_annotations\_in\_either}$

Term Use: Common Terms

Threshold: .5

Maximum number of deselections: 2

Minimum number of common terms: 0

The document pair initially selected was documents A and B.

Document selection algorithm steps are outlined as follows:

Candidate Pair A,C

$\text{ADiff}_A = 0.0$ $\text{ADiff}_C = 0.0$

No deselection

Candidate Pair A,B

$\text{ADiff}_A = 0.125$ $\text{ADiff}_B = 0.25$ $\text{ADiff}_C = 0.125$

No deselection

Candidate Pair C,D

$\text{ADiff}_A = 0.25$ $\text{ADiff}_B = 0.5$ $\text{ADiff}_C = 0.183$ $\text{ADiff}_D = 0.587$

Deselect D Count = 1

Candidate Pair C,D

$\text{ADiff}_A = 0.25$ $\text{ADiff}_B = 0.5$ $\text{ADiff}_C = 0.183$ $\text{ADiff}_D = 0.587$

Deselect D Count = 2

The document selection algorithm results in Documents A, B and C remaining in the set for further analysis. Annotated and inferred terms from the annotations were used in the similarity calculations. It should be noted that the actual value of the ontological similarity measure was not as important as is the relationship between the similarity measures and the common term indicators of textual similarity.

To calculate the WHIsim measure, it was necessary to find the index of the lowest parent of each concept. For example, Concept 1 “Text Mining” has “Artificial Intelligence” as its lowest parent. There were 7 concepts indicated for Document A. Of those seven, three had Concept 8 as the lowest parent and four had Concept 9. The weight for the concepts with lowest parent 8 will be 4/7, and likewise the weight for those with Concept 9 as lowest parent will be 3/7. Based on the definition of WHIsim and the formula defined in Equation 3.8, the calculations for the document pair A, B was:

$$WHIsim(d_a\, d_b,) = \frac{(0\times 0)+((3/7)\times 0)+((4/7)\times(4/4))+(0\times 0)+(0\times 0)}{\sqrt{(0+(3/7)^2+(4/7)^2+0+0}\times\sqrt{(0+0+(4/4)^2+0+0}}$$

$$= \frac{4/7}{\sqrt{(0+(9/49)+(16/49)\ \ +0+0}\times\sqrt{1}}$$

$$= \frac{4/7}{\sqrt{(25/49)}\times 1}$$

$$= \frac{4/7}{5/7} = .8$$

In order to implement the similarity metrics suggested by Ganesan, Garcia-Molina and Widom (2003), it is helpful to view the ontology in a tree form. Each of the concepts of interest must be represented as a node in the tree. In the SWAS tool, a list was maintained of the ancestors and children of each node. This enabled calculations to be based on the node itself, or on the ancestors of the node as suggested by Ganesan, Garcia-Molina and Widom (2003), or to group nodes with common children, as suggested by Ehrig and Sure (2004). The example presented here uses only 11 of 47 possible concepts in the ontology. The structure specified by the authors requires that those instances that will be considered in the similarity analysis (concepts of interest) be modeled as leaf nodes of the tree. Many terms could be both specified instances as well as superclasses for other instances, which placed those as interior nodes of the tree. For the purpose of this analysis, each interior node was considered both a leaf and a superclass. Computationally in the SWAS system, a list was maintained of the ancestors and children of each node, as well as the level of the tree where the node appears. In addition, the lowest ancestor of each node was stored. This information was used in the new similarity measures that were proposed. Figure 8 shows the relevant terms from the ontology as a tree structure. The node “Artificial Intelligence” was be the lowest ancestor of “Text Mining”, “Knowledge Discovery” and “Agents”.

.

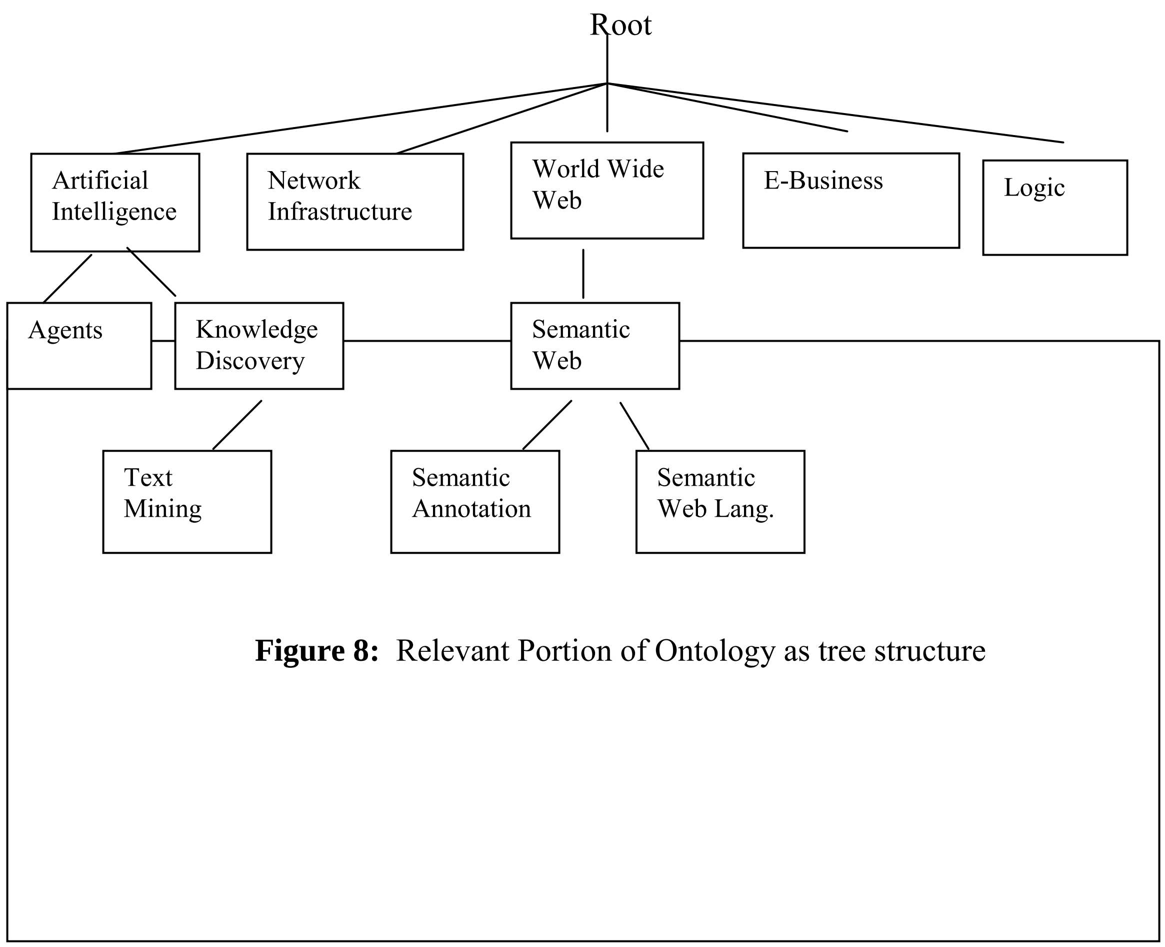


Figure 8: Relevant Portion of Ontology as tree structure

The three documents contained the following concepts of interest:

A = (Text Mining (*l_1*), Semantic Annotation(*l_2*), Semantic Web Languages(*l_3*))

B = (Semantic Annotation(*l_2*), Semantic Web Languages(*l_3*))

C = (Agents(*l_4*), Logic(*l_5*), Text Mining(*l_1*), Semantic Annotation(*l_2*))

Using the notation suggested by Ganesan, Garcia-Molina and Widom (2003), the similarities were calculated as follows (note, the count of instances represented by a_i b_j has been omitted, since it is a value of 1 which serves as a multiplier):

$$\vec{l_1} \cdot \vec{l_2} = \frac{2 * depth(LCA_U(l_1 \cdot l_2))}{depth(l_1) + depth(l_2)} = \frac{2*1}{4+4} = 0.25$$

$$\vec{l_1} \cdot \vec{l_3} = \ = \frac{2*1}{4+4} = 0.25$$

$$\vec{l_1} \cdot \vec{l_4} = \ = \frac{2*2}{4+3} = 0.57$$

$$\vec{l_1} \cdot \vec{l_5} = \ = \frac{2*1}{4+2} = 0.33$$

$$\vec{l_2} \cdot \vec{l_3} = \ = \frac{2*3}{4+4} = 0.75$$

$$\vec{l_2} \cdot \vec{l_4} = \ = \frac{2*1}{4+3} = 0.29$$

$$\vec{l_2} \cdot \vec{l_5} = \ = \frac{2*1}{4+2} = 0.33$$

$$\vec{l_3} \cdot \vec{l_4} = \ = \frac{2*1}{4+3} = 0.29$$

$$\vec{l_3} \cdot \vec{l_5} = \ = \frac{2*1}{4+2} = 0.33$$

$$\vec{l_4} \cdot \vec{l_5} = \; = \frac{2*1}{3+2} = 0.4$$

$$\vec{A} \cdot \vec{B} = \sum_{i=1}^{nA} \sum_{j=1}^{nB} \cdot \vec{l_i} \cdot \vec{l_j} = 0.25 + 0.25 + 1 + 0.75 + 0.75 + 1 = 4.0$$

$$\vec{A} \cdot \vec{C} = \quad = 1 + 0.25 + 0.57 + 0.33 + 0.25 + 1 + 0.29 + 0.33 + 0.25 + 0.75 + 0.29 + 0.33 = 4.02$$

$$\vec{B} \cdot \vec{C} = \quad = 0.25 + 1 + 0.29 + 0.33 + 0.25 + 0.75 + 0.29 + 0.33 = 3.49$$

$$\vec{A} \cdot \vec{A} = \quad = 1 + 0.25 + 0.25 + 0.25 + 1 + 0.75 + 0.25 + 0.75 + 1 = 5.5$$

$$\vec{B} \cdot \vec{B} = \quad = 1 + 0.75 + 0.75 + 1 = 3.5$$

$$\vec{C} \cdot \vec{C} = \quad = 1 + 0.25 + 0.57 + 0.33 + 0.25 + 1 + 0.29 + 0.33 + 0.57 + 0.29 + 1 + 0.4 + 0.33 + 0.33 + 0.4 + 1 = 7.34$$

$$GCSM\ sim\ (A, B) = \frac{\vec{A} \cdot \vec{B}}{\sqrt{\vec{A} \cdot \vec{A}}\sqrt{\vec{B} \cdot \vec{B}}} = 0.91$$

$$GCSM\ sim\ (A, C) \quad \frac{\vec{A} \cdot \vec{C}}{\sqrt{\vec{A} \cdot \vec{A}}\sqrt{\vec{C} \cdot \vec{C}}} = 0.63$$

$$GCSM\ sim\ (B, C) = \frac{\vec{B} \cdot \vec{C}}{\sqrt{\vec{B} \cdot \vec{B}}\sqrt{\vec{C} \cdot \vec{C}}} = 0.69$$

Calculations for the second measure proposed by Ganesan, Garcia-Molina and Widom (2003) were computed to demonstrate this method. This measure compared the leaf similarity in each collection, assuming an induced tree was formed to show the leaf nodes specified in each collection. The similarity measures were asymmetric and an average of the similarity value for A and B would have to be computed to find a true symmetric measure. This is illustrated below for the calculation of the symmetry between document A and B.

The relevant leaf nodes could be identified as:

l_1 = Text Mining

l_2 = Semantic Annotation

l_3 = Semantic Web Languages

l_4 = Agents

l_5 = Logic

First, the calculations to determine the similarity of Document A to Document B:

match $_{TA,\,TB}$ = *LCA* T_A, T_B (l_1) = Topic.

$$leafsim_{TA,TB}(l_1) = \frac{depth(LCA_{TA,TB}(l_1))}{depth(l_1)} = 1/3 = .33$$

match $_{TA,\,TB}$ = *LCA* T_A, T_B (l_2) = Semantic Annotation

$$leafsim_{TA,TB}(l_2) = \frac{depth(LCA_{TA,TB}(l_2))}{depth(l_2)} = 4/4 = 1$$

match $_{TA,\,TB}$ = *LCA* T_A, T_B (l_3) = Semantic Web Languages

$$leafsim_{TA,TB}(l_3) = \frac{depth(LCA_{TA,TB}(l_3))}{depth(l_3)} = 4/4 = 1$$

$$OGM\ sim(A,B) = \sum_{l1 \in A} leafsimT_1, T_2(l_i)/n = 2.33\ /\ 3 = .78$$

Next, the calculations were repeated to find the similarity of Document B to Document A:

match $_{TB,\,TA}$ = *LCA* T_B, T_A (l_2) = Semantic Annotation

$$leafsim_{TB,TA}(l_2) = \frac{depth(LCA_{TB,TA} l_2))}{depth(l_2)} = 4/4 = 1$$

$match_{TB,\,TA} = LCA\ T_B,T_A\ (l_3)$ = Semantic Web Languages

$leafsim_{TB,TA}(l_3) = \dfrac{depth(LCA_{TA,TB}(l_3))}{depth(l_3)} = 4/4 = 1$

$OGM\ sim(B,A) = \sum_{l1 \in A} leafsimT_1, T_2(l_i)/n = 2 / 2 = 1$

Using the average of the asymmetric measures, the symmetric measure of similarity between A and B is 1.78 / 2 = .89. In a similar way, the symmetric measures can be calculated for the similarity between documents A and C (.93) and documents B and C (.51). Due to the asymmetric nature of this calculation, and the fact that the subclass structure necessitates the assumption that interior nodes are considered as both leaf nodes and interior nodes, a decision was made to not include this measure in the final SWAS analysis.

As can be seen by the variety and results of similarity calculations, the meaning of these values is dependent on the context in which they are viewed. In this case, document annotation similarity measures were examined in conjunction with the common terms and common unexpected terms (implicit knowledge) found in the text of the documents themselves. Exploring the unexpected terms involves examination of the vocabulary of each document. Reverse stemming techniques, implemented through suffix elimination, was used to determine roots of the terms in the documents, and comparisons were made to the root of the terms rather than the term itself. This type of term grouping had been previously employed by Liu, Ma and Yu (2000) and others.

After the common term roots and common unexpected term roots were determined, each of these sets of measurements was correlated to each similarity measure to estimate the best measure of similarity. Table 12 shows the common term roots and

CURs found from the analysis. Table 13 shows the value of each similarity measure, along with the number of common terms and CURs and the correlation as computed by the Pearson correlation coefficient. Table 13 shows the same data based on rankings, utilizing Spearman's rank correlation coefficient.

Document Pair	**Common Terms (except CURs)**	**CURs**
A, B	Semantic Web	Designed Documents Enhanced
A, C	Semantic Web	Add Communities Use, Users
B, C	Knowledge Web	Enable

Table 12: Common Terms and Unexpected Terms in Example Documents

Doc. Pair	Common Terms	Curs	Ssim	Jsim	Vsim	UHIsim	WHIsim	GCSM
A,B	5	3	.82	.67	.14	.5	.8	0.91
A,C	5	3	.64	.56	.10	.33	.75	0.63
B,C	3	1	.45	.33	.09	.33	.12	0.69
Common Term Corr.			0.87	0.95	0.65	0.50	1.0	0.31
CUR Corr.			0.87	0.95	0.65	0.50	1.0	0.31

Table 13: Example Summary of Common Terms, Measures and Correlation

Doc. Pair	Common Terms	Curs	Rank Ssim	Rank Jsim	Rank Vsim	Rank UHIsim	Rank WHIsim	Rank GCSM
A,B	2.5	2.5	3	3	3	3	3	3
A,C	2.5	2.5	2	2	2	1.5	2	1
B,C	1	1	1	1	1	1.5	1	2
Common Term Corr.			.13	.13	.13	.38	.13	.88
CUT Corr.			.13	.13	.13	.38	.13	.88

Table 14: Example Ranking of Common Terms, Measures and Correlation

In the particular example shown, the correlations for common terms and CURs are the same, but this is coincidental and not expected to be the case when actual documents are analyzed. The correlations in Table 12 were calculated by the Pearson correlation coefficient, and those in Table 13 were calculated by the Spearman ranked correlation coefficient, as shown in Equations 2.22 and 2.23. The actual data also showed the effect of using the inferred concepts for the non-hierarchical measures, although those results are not shown here to keep the example brief.

The approach used was based on the premise that terms indicated via the annotation ontology were actually part of the document annotation. For example, “Text Mining” was the first concept specified in the ontological annotation to be considered in the documents. In Document C, the third sample document created, the topic annotation “Text Mining” was included. While neither word contained in “text mining” was a part of the document text, the ontology defines the concept “Knowledge Discovery” as the parent of text mining and since “knowledge” was included in the document text, then the data reflects that this term is inferred for the third document. In other words, the term “knowledge” was inferred from the annotation “Text Mining”.

No conclusions were drawn from this sample data for two reasons. First, the sample size of four documents was too small to yield any reliable results. In addition, although an actual published ontology was used for the semantic markup, the documents themselves were contrived to fit this example, rather than actual data already published on the Web. This example pointed out one problem that would have to be addressed by the Webmaster of a Web portal. If a tool to compare similarity of annotations and document content were to be placed on the portal, it would be extremely important to validate the reliability of the documents placed there that others in the community would be measuring against. Algorithms such as the document selection technique used in this study could be appropriate for this task.

In summary, the following steps were followed in this example:

1. Select the documents and ontology to be utilized. Specify the concepts that will be included, as well as the inference rules that shall be applied for that ontology. Further refine the set of selected documents according to the threshold and deselection criteria.
2. Create the ontological similarity document term matrix and indicate explicit and inferred terms.
3. Calculate the similarity of each document pair with various measures.
4. Relate all terms to a term root. Prune the text portion of the documents to remove common non-descriptive terms using a stop-list. Create a vocabulary matrix that includes all non-pruned terms for each document. Enter the frequency of each term in each document, as well as a flag indicating if this term is unexpected based on the ontological annotation and inference.
5. For each document pair, identify the common term roots and the common unexpected term roots.
6. Compare the various similarity values to the number of common unexpected term roots and common term roots for each document pair. Determine if there is a positive correlation between these two measures.

The same process of steps was carried out utilizing the SWAS tool to examine actual documents with the full ontology. Multiple parameters were varied, and the results analyzed according to those various criteria. Findings were formulated, which led to suggestions for future research.

Chapter 4

Results

Findings

Utilizing the Semantic Web Analysis of Similarity (SWAS) tool, parametric studies were conducted to evaluate the correlation of the various measures of similarity to the number of common term roots and common unexpected term roots in the document set. The correlation coefficients were then analyzed to determine if they were statistically significant. A point value was assigned to the significant measures, with the highest measure receiving the most points, and the non-significant measures receiving the least points. Utilization of the point system, based on the ranking of each measure as compared to the others, allowed comparison of the various measures grouped by the various parametric criteria.

The online tool available at http://faculty.vassar.edu/lowry/ch4apx.html was used to determine the statistical significance. Since only positive correlation was desired, a one-tailed or directional test of significance was applied. Those values not reaching a 5% level of significance were deemed insignificant, since this value is commonly used for scientific research (Lowry, 2007). This minimum value was calculated for all the sample sizes evaluated, and is shown as the "minimum rho" value in the data tables.

Eighteen measures of similarity were computed in the SWAS analysis. The similarity measures are labeled M_1 through M_{18} in the data tables. Table 15 shows the meaning of each of these measures.

Label	Measure
M_1	Simple Match Measure
M_2	Ranked Simple Match Measure
M_3	Simple Match Measure using Inferred Concepts
M_4	Ranked Simple Match Measure using Inferred Concepts
M_5	Jacquard Measure
M_6	Ranked Jacquard Measure
M_7	Jacquard Measure using Inferred Concepts
M_8	Ranked Jacquard Measure using Inferred Concepts
M_9	Cosine Measure
M_{10}	Ranked Cosine Measure
M_{11}	Cosine Measure using Inferred Concepts
M_{12}	Ranked Cosine Measure using Inferred Concepts
M_{13}	GCSM Measure using Inferred Concepts
M_{14}	Ranked GCSM Measure using Inferred Concepts
M_{15}	UHI Measure using Inferred Concepts
M_{16}	Ranked UHI Measure using Inferred Concepts
M_{17}	WHI Measure using Inferred Concepts
M_{18}	Ranked WHI Measure using Inferred Concepts

Table 15: Similarity Measure Legend

The measure with the highest significant positive correlation was assigned a value of 18 points and the measure with the next highest significant positive correlation was assigned a value of 17 points, and so on until all measures with significant positive correlation were assigned a point value. Those measures with negative or insignificant correlation received no point value. This allowed various parametric runs to be grouped together and results compared.

The Generalized Cosine-Similarity Measure (GCSM), unweighted hierarchical measure (UHI), and weighted hierarchical measure (WHI) made use of the hierarchical structure of the ontology. and were built on the premise of inclusion of inferred concepts, so these measures have little meaning without considering the inferences. Rule 1 (Formal Language) must be considered in the context of the ontology. There are ten formal languages specified, but there are no subclasses for any of them. Since each annotated concept is at the same level of the ontology, and the GCSM measure was designed to show the relationship based on the depth from a common ancestor, this measure was not useful when considering this rule. For a similar reason, there was no difference in the results when using docRatio or parentRatio if the validity of annotation was determined by Rule 1 only. There was also no difference between the annotations and inferred measures when Rule 1 was used (since there is no inferred data).

Valid annotations were determined based on the rules. Selection sets were based on the docRatio and parentRatio separately. The number of deselections was held constant at five, since this resulted in a selection set that could not be improved further by more deselections. Correlation was measured for the common term roots and common unexpected term roots. Five threshold values were examined for each of these cases, corresponding to a sample size of approximately one to three percent of the total documents, and one-quarter, one-half, three-quarters, and all of the documents. In some cases, a minute difference in the threshold value caused a major change in the document selection, and in other cases, a wider range of thresholds yielded no change in the document selection. At least ten documents were included in the selection set for all cases analyzed. The correlation required to be considered significant varied inversely with the sample size. Small samples require a high correlation coefficient to be considered significant, as compared to the value required for large sample sizes.

Table 16 shows an example of part of the comparison for the threshold value of 0.128 examined using Rule 2 when the selections were based on the docRatio and the correlations were based on common term roots. Only document pairs with at least one common term were considered for inclusion in the selection set. This threshold yielded 28 document pairs in the selection set, and those correlations with a value more than 0.317223 were considered significant.

Rule 2 Topic MinCommon1 Thresh 0.128 for Doc CR							
Doc1	**Doc2**	**Com Roots**	**Rank CR**	**M_5**	**M_6**	**M_7**	**M_8**
6	7	3	12	0.27273	4.5	0.31429	6
6	10	4	18.5	0.6	19	0.55	15.5
6	14	3	12	0.23077	3	0.275	5
6	28	6	26.5	0.375	10.5	0.6	17
6	33	6	26.5	1	27	0.78571	18.5
6	61	2	5	0.75	23.5	0.88	24
6	62	4	18.5	0.66667	21	0.8	20.5
7	10	3	12	0.33333	7.5	0.18333	2
7	14	3	12	1	27	1	27.5
7	28	0	1	0.5	16	0.44	11.5
7	33	5	22.5	0.75	23.5	0.91667	25
7	61	3	12	0.42857	13.5	0.44	11.5
7	62	5	22.5	0.375	10.5	0.4	10
10	14	2	5	0.27273	4.5	0.15714	1
10	28	1	2	0	1.5	0.22	4
10	33	5	22.5	0.75	23.5	1	27.5
10	61	2	5	0.42857	13.5	0.48889	13
10	62	2	5	0	1.5	0.2	3
14	28	3	12	0.375	10.5	0.36667	8.5
14	33	6	26.5	0.6	19	0.78571	18.5
14	61	4	18.5	0.33333	7.5	0.36667	8.5
14	62	3	12	0.3	6	0.33846	7
28	33	5	22.5	0.5	16	0.55	15.5
28	61	2	5	0.75	23.5	0.94286	26
28	62	3	12	0.6	19	0.825	22.5
33	61	6	26.5	1	27	0.8	20.5
33	62	4	18.5	0.375	10.5	0.50769	14
61	62	3	12	0.5	16	0.825	22.5
Corr.				**0.43546**		**0.38589**	
Rank Corr.					**0.40476**		**0.3685**
Number of pairs selected=			28				

Table 16: Example SWAS Output

The four measures associated with the Jacquard measure are shown in this table (Measures M_5-M_8). From those four measures, all of the correlation values were significant, and those are shown in bold in Table 16. Full results for all measures and all parametric cases are available at http://home.cfl.rr.com/lookhome/joelynn/SWASoutput. The summary data is also available at that location.

Table 17 shows a portion of the summary data for all thresholds for this same case. Summaries of other cases are shown in Appendix D. The column in Table 17 labeled “Min rho” refers to the smallest value of the correlation coefficient that was considered significant based on a one-tailed or non-directional test of significance.

Correlation and Ranked Correlation Rule 2 MinCommon 1 Document Ratio Common Roots								
(Significant Correlations Shown in Bold)								
Threshold	#PrsSel	Min rho	MaxVal	MaxMeas	M5	M6	M7	M8
0.128	28	0.317	0.435	JMatch	**0.435**	**0.405**	**0.386**	**0.369**
	Rank				1	4	8	9
	Points				18	15	11	10
0.155	561	0.070	0.340	RankJMatch	**0.264**	**0.340**	**0.202**	**0.241**
	Rank				4	1	9	5
	Points				15	18	10	14
0.175	1176	0.048	0.242	RankJMatch	**0.183**	**0.242**	**0.141**	**0.148**
	Rank				7	1	11	9
	Points				12	18	8	10
0.218	1770	0.039	0.163	RankJMatch	**0.137**	**0.163**	**0.131**	**0.102**
	Rank				6	1	7	11
	Points				13	18	12	8
1.000	2346	0.034	0.114	RankJMatch	**0.069**	**0.114**	**0.059**	**0.041**
	Rank				5	1	7	10
	Points				14	18	12	9
	Total Pts			RankJMatch	72	87	53	51
	Rank TP				4	1	6	7

Table 17: Sample Summary Analysis

The data was grouped in order to analyze not only the performance of the various measures, but also the results of each parametric variation. The data was grouped according to rule, ratio for selection, common term roots or common unexpected term roots, and whether or not document pairs in the selection set have at least one common term. The measures themselves denoted whether terms derived from knowledge that could be inferred from the ontology were considered as annotated, and whether the correlation was based on normed values or on the rankings of the values.

Summary of Results

The results from this experiment validated the results obtained from Haveliwala, Gionis, Klein and Indyk in 2002. Over all the cases studied, the Jacquard measure showed the highest correlation when compared to either the common term roots or the common unexpected term roots. This measure, essentially the union of the common terms divided by the intersection, not only best correlated to the common terms, but was computationally simple and easy to understand. Since a primary aim of this study was to build a foundation for tools for novices Semantic Web users, the appropriateness of a common and widely understood measure was an added bonus. Four different cases were analyzed for most of the measures, including the Jacquard measure. The similarity measure was calculated based on the annotated ontological concepts and on all concepts that could be inferred from the ontology. The correlation was calculated based on the actual values and also on the relative ranking of those values. The measure that used the rankings of values rather than the values themselves resulted in higher correlation for the Jacard measure.

The results of the data remained reasonably consistent no matter which grouping was used. Table 18 gives an overview of the results, showing the rankings of all measures for a selection of the groupings. A ranking of 1 meant that the measure was ranked the highest, and a ranking of 18 denoted the lowest ranking.

	All rules	**Rule 2**	**Rule 1**	**Doc Ratio**	**Parent Ratio**	**Com Roots**	**CUR**	**Min Com**	**Min Com**
M_1	18	17	13	17	15	17	18	18	18
M_2	15	15	13	15	12	16	12	15	15
M_3	17	18	13	17	14	15	17	17	17
M_4	16	16	13	16	16	18	16	16	16
M_5	3	5	7	3	3	3	3	3	3
M_6	1	1	1	1	1	1	1	1	1
M_7	6	6	7	7	7	6	6	6	6
M_8	7	7	1	6	8	7	7	7	7
M_9	4	4	9	4	5	5	4	5	4
M_{10}	2	2	3	2	2	2	2	2	2
M_{11}	11	10	9	12	10	10	13	10	12
M_{12}	9	9	3	9	9	8	9	9	9
M_{13}	13	12	13	14	17	13	15	13	14
M_{14}	14	13	13	13	18	14	14	14	13
M_{15}	8	8	9	10	6	9	8	8	8
M_{16}	5	3	3	5	4	4	5	4	5
M_{17}	12	11	12	11	11	11	11	11	11
M_{18}	10	14	3	8	13	12	10	12	10

Table 18: Performance of all Measures

While there is little variance in the results, it is obvious that the least consistent group is when Rule 1 is used alone. Because there were no superclass/subclass designations in this portion of the ontology, this rule skewed the results. Rule 1 was deemed inappropriate for comparison to the hierarchical results.

M_6, the ranked Jacquard measure based on the annotations only, amassed the highest point value in each grouping. When considering only Rules 2 and 3, the highest point value for that measure was obtained using Rule 3, which included annotations that were valid according to either the formal language or the topic criteria. Table 19 shows the average value of M_6 for various groupings using Rule 3.

Ranked Correlation Values for Jacquard Measure Rule 3	
Grouping	**Points for M6**
DocRatio	333
ParentRatio	345
Common Roots	338
Common Unexpected Roots	340
MinCom 0	326
MinCom 1	352

Table 19: Point Values for M_6 for Various Groupings

Examination of this grouped data shows that a higher point value for this measure is amassed when the parentRatio is used for document selection and when only those documents with at least one common term are considered. The point value when correlating to common unexpected term roots is higher than when the common terms are used, but this difference is slight.

One further aggregation of the data for Rule 2 was done to show the overall effect of each similarity measure. There were six basic measures considered (Simple Match, Jacquard, Cosine, GCSM, UHI and WHI). For the first three of these measures, there were four variations considered (measure based on value of correlation without inferred concepts, measure based on ranking of correlation without inferred concepts, and the value and ranking when inferred concepts were included). Since the three hierarchical measures depended on the inferred concepts, only the value of correlation and the ranked value were shown. All points for each measure were totaled, and those totals doubled for the hierarchical measures, since only half as many variations were considered.

This data is shown in Table 20, and was grouped to show the cases where the selection set was determined by the docRatio value in Table 21 and the parentRatio value in Table 22. The final two tables show the effect of the selection criteria on the overall performance of that measure. It was reasonable to assume that the hierarchical measures would perform better when the document selection was based on parentRatio, since that calculation took into effect the hierarchical structure of the ontology, and Table 22 does show that the UHI hierarchical measure is the highest ranked one. Only rule 2 was used for this portion of the analysis, since Rule 1 was based on a part of the ontology that had no hierarchical structures, and Rule 3 included the results from Rule 1.

Points for Rule 2 Aggregated for Each Measure								
Rule	MinCom	Case	SimpleM	Jacquard	Cosine	GCSM	UHI	WHI
2	1	DocCR	0	263	240	156	214	146
2	1	DocCUR	33	172	109	154	116	66
2	1	ParCR	0	263	255	12	284	150
2	1	ParCUR	111	155	152	0	138	0
2	0	DocCR	0	254	242	156	224	96
2	0	DocCUR	33	203	141	188	116	66
2	0	ParCR	0	263	255	12	284	150
2	0	ParCUR	111	155	152	0	138	0
		Total	**288**	**1728**	**1546**	**678**	**1514**	**674**
		Rank	**6**	**1**	**2**	**4**	**3**	**5**

Table 20: Rule 2 Data Aggregated for Each Measure

Points for Rule 2 Aggregated for Each Measure docRatio Selection								
Rule	MinCom	Case	SimpleM	Jacquard	Cosine	GCSM	UHI	WHI
2	1	DocCR	0	263	240	156	214	146
2	1	DocCUR	33	172	109	154	116	66
2	0	DocCR	0	254	242	156	224	96
2	0	DocCUR	33	203	141	188	116	66
		Total	**66**	**892**	**732**	**654**	**670**	**374**
		Rank	**6**	**1**	**2**	**4**	**3**	**5**

Table 21: Rule 2 Aggregated Data Based on DocRatio Selection

Ranking Summary Rule 2						
Aggregate	SimpleM	Jacquard	Cosine	GCSM	UHI	WHI
All Cases	6	1	2	4	3	5
DocRatio	6	1	2	4	3	5
ParentRatio	5	2	3	6	1	4

Table 22: Rule 2 Aggregated Data Based on ParentRatio Selection

Chapter 5

Conclusions, Implications, Recommendations and Summary

Conclusions

Based on the analysis performed, the best measure of similarity for this data set is the Jacquard measure when the ranked values are compared to the common unexpected terms documents selected using the parentRatio. Examination of this grouped data shows that a higher point value for this measure is amassed when the parentRatio is used for document selection and when only those documents with at least one common term are considered. The point value when correlating to common unexpected term roots is higher than when the common terms are used, but this difference is slight. It should be noted that these conclusions are based on documents annotated from a single ontology, and any generalization of these conclusions should not be made without further study.

The results demonstrate that the area of document selection is important. When the data was aggregated into the six basic measures and compared by the method of document selection, and when when the document selection was based on the parentRatio calculation, the highest ranked measure became the unweighted hierarchical measure (UHI). It would be expected that the hierarchical measures would perform better when selection was based on parentRatio, since this took into account the hierarchical structure of the ontology. Only one method of automated document selection with two variations was presented in this study. Other studies are needed to compare this method to other selection algorithms, as well as to selection by knowledge management experts.

While none of the hierarchical measures of similarity were the highest ranked for the individual variations of the measures, this study did show some benefit of the knowledge provided through the ontological knowledge. The use of the parentRatio to implement the selection algorithm is only possible through the definition of the concept tree provided in the ontology. Determination of which terms were unexpected was also possible only because of the knowledge derived from the ontology. This study shows that there is promise for the knowledge gained from the ontology in determination of document similarity, but the quantification of its usefulness should be determined only after further study with a broader range of ontologies and document sets.

Implications and Recommendations

Based on the results of this study, recommendations were formulated that fell into three broad categories. These were improvement to the SWAS system, extended studied of similarity measurements and document selection involving other ontologies or document sets, and development of web portal tools to assist with semantic annotation. The need for ongoing research has been shown through the development of the Semantic Web to this point.

The current SWAS system was built as a prototype for preliminary studies of similarity measures applied to SWDs. A significant amount of pre-processing and post-processing was done off-line to prepare input and analyze output files. The InputParamters file was designed as a text-based input file that required the user fully understand the system implementation to correctly configure. This file required that the user specify maximum numbers concerning the ontology and document set to work properly. Incorporation of methods to extract the ontological markups and parse those into RDF triples would eliminate much of the pre-processing work. Conversion of the array-based implementation of the code to a vector-based implementation would eliminate the need to specify maximum values. Development of a graphical user interface would simplify the parametric input. Post-processing methods built into the system would reduce the off-line analysis required. Source code has been uploaded to facilitate enhancements to this system by interested researchers.

An analysis of the algorithms used for searching and sorting throughout the SWAS system could lead to other implementations of the same tasks that improve performance. Scalability concerns were not addressed in this initial implementation. This remains an area for future research. Scalability should be studied if this tool is to be used with large collections of documents. In addition, further statistical studies on the similarity measures may prove to be of interest. Computationally complex models such as those used by the SMART system were not included in this initial analysis.

Other ways the SWAS system could be improved include enhancing the text analysis by incorporating more elaborate means of determining term roots, and incorporation of synonyms for those roots and expansion into languages other than English. Incorporation of online dictionary tools, such as WordNet, to determine term roots would be a logical next step in the refinement of this tool. Incorporation of synonyms of terms would also be a major refinement, as would incorporation of translators that would allow implementation in other languages. It should be noted that not only would data have to be translated, but the stop list as well.

Research conclusions cannot be based on a single study. The results of this study did confirm results of the 2002 study performed by Haveliwala, Gionis, Klein and Indyk investigating similarity measures in a different context. More research is needed to analyze various ontologies and annotations before conclusions can be drawn. Reseach is also needed to determine selection algorithms to narrow a document set to those most appropriate for the analysis. Use of automated techniques such as the docRatio and parentRatio methods utilized by SWAS should be compared to results of a knowledge expert to determine the feasibility of automated selection. While the Jacquard measure appeared to be appropriate for this ontology and document set, other measures would likely be a better fit for different ontologies.

This study showed that automated systems could be used to select documents and determine similarity to an ontology. This can be extended to provide validation tools for semantic markup. One reason frequently attributed to the success of the World Wide Web is that it is intuitive and simple for users. One problem cited with the Semantic Web is that significant knowledge engineering background is needed to produce SWDs with true meaning encoded in the annotations. The lack of tools to assist users with this process has been cited as one reason the Semantic Web has not become more popular (Haustein & Pleumann, 2002). One of the primary use cases suggested for the Semantic Web is that of a portal for users of a common ontology. As such portals are developed, groups of users will likely display documents in a general area of interest.

Currently, there is no real benefit to users who semantically enhance documents. If portal tools were developed to search document databases based on semantic annotations, users who produced documents with those annotations would have the benefit of an additional way for others to find that document. Key to the idea of communities of users is the idea that simple tools can be developed to assist users in determining and validating semantic information added to the documents. Novice web authors do not have the knowledge engineering background to discern most appropriate annotations for a specific document. Semantic similarity could be used as a method for validation of annotations. Once a set of appropriately annotated documents was selected, automated tools could be developed to allow the novice user to submit either an annotated document or the document text only.

If the user submitted the document text only, and there was a sufficient supply of appropriately annotated documents available within the Web portal, the text of the document could be compared to the text of those already deemed suitable. The similarity measure shown to be most appropriate for this ontology could be utilized to determine which document was most similar to this one. The annotations used for the most similar document could be suggested to this user. The results of the text comparison could be used to display to the user how several similar documents were annotated.

If the annotations were submitted, the system could supply a measure indicating how this document compares to others with similar annotations. By utilizing the same procedure that was applied in the document selection scheme in SWAS system and outlined in the “Selection Algorithm” of chapter 3 of this document, an average selected difference could be calculated for the documents in the pre-determined set of appropriately annotated documents, and this value recalculated to include the user’s document. If the average difference of the document set including that document did not significantly decrease, then the document would be considered appropriately annotated. This information could be transmitted back to the user, along with annotation suggestions based on the document text alone if the document were determined inappropriately annotated for this community of users.

Summary

The SWAS tool has been developed to provide a means for comparison of document text. Source code for the tool has be published online, allowing free use of this tool for enhancements and future research. Detailed descriptions of the data preprocessing and operating system setup have been provided. The data analysis showed the Jacquard measure most appropriate for the overall data studied, and the unweighted hierarchical measure a good choice if only the topic of the paper is used as the criteria for concepts to be used from the ontology and the parentRatio method of document selection employed. A suggestion has been presented for the development of a Web portal tool to provide novice Semantic Web users with information concerning annotations from documents with similar text.

Appendix A

Source Code Listing of Sample Documents

Document A:

```
<!DOCTYPE html PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<!-- NOTE:  This is an example of an annotated document using the ISWC ontology
    created to demonstrate computation of ontological similarity -->
<html>
    <head>
        <!--<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                    xmlns:daml=http://www.daml.org/2001/03/daml+oil#"
                    xmlns="http://annotation.semanticweb.org/iswc/iswc.daml#">
        <InProceedings
            rdf:about="http://home.cfl.rr.com/lookhome/joelynn/pdp/ResourceC.html#
                Overview of the Semantic Web">
            <conference rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                ISWC_2002"/>
            <title>   Uses of Semantic Web Documents   </title
            <topic  rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Annotation"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Web_Languages"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Text_Mining"/>
            <year>    2003    </year>
        </InProceedings>
        </rdf:RDF>    -->
    </head>
    <body>
        <h2>   Uses of Semantic Web Documents   </h2>
        <p>Documents designed for the Semantic Web can be used  to enhance the
            performance of search engines, add more  understanding to the data, and provide
            information to communities of users on the Web.   </p>
    </body>
</html>
```

Document B:

```
<!DOCTYPE html PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<!-- NOTE:  This is an example of an annotated document using the ISWC ontology
    created to demonstrate computation of ontological similarity -->
<html>
    <head>
        <!-- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
                    xmlns="http://annotation.semanticweb.org/iswc/iswc.daml#">
        <InProceedings
            rdf:about="http://home.cfl.rr.com/lookho1me/joelynn/pdp/ResourceC.html#
                Overview of the Semantic Web">
            <conference rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                ISWC_2002"/>
            <title>   Languages of the Semantic Web   </title>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Web_Languages"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Annotation"/>
            <year>    2003    </year>
        </InProceedings>
        </rdf:RDF>
         -->
    </head>
    <body>
        <h2>      Languages of the Semantic Web    </h2>
        <p>A hierarchy of languages support the development  of the Semantic Web
            according to standards being  developed by W3C. These languages, including
            XML  and RDF,  enable the addition of knowledge  which can be understood by
            machines to documents on the Web. This is accomplished through the use of   an
            ontology which has been designed for that particular subject, and used to
            enhance the Web documents.     </p>
    </body>
</html>
```

DocumentC :

```
<!DOCTYPE html PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<!-- NOTE:  This is an example of an annotated document using the ISWC ontology
    created to demonstrate computation of ontological similarity -->
<html>
    <head>
        <!-- <rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
            xmlns:daml=http://www.daml.org/2001/03/daml+oil#
            xmlns="http://annotation.semanticweb.org/iswc/iswc.daml#">
        <InProceedings
            rdf:about="http://home.cfl.rr.com/lookho1me/joelynn/pdp/ResourceC.html#
                Overview of the Semantic Web">
            <conference rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                ISWC_2002"/>
            <title>   Overview of the Semantic Web   </title>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Agents"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Logic"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Text_Mining"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Annotation"/>
            <year>   2003   </year>
        </InProceedings>
        </rdf:RDF>
         -->
    </head>
    <body>
    <h2>   Overview of the Semantic Web    </h2>
    <p>The Semantic Web is a project being worked on by Tim Berners-Lee. It will add
        logic to the Web, and enable agents to infer knowledge and concepts that extend
        beyond keywords for use by agents and communities of users.    </p>
    </body>
</html>
```

Document D:

```
<!DOCTYPE html PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<!-- NOTE:  This is an example of an annotated document using the ISWC ontology
    created to demonstrate computation of ontological similarity -->
<html>
    <head>
        <!-- <rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
            xmlns:daml=http://www.daml.org/2001/03/daml+oil#
            xmlns="http://annotation.semanticweb.org/iswc/iswc.daml#">
        <InProceedings
            rdf:about="http://home.cfl.rr.com/lookho1me/joelynn/pdp/ResourceC.html#
                Overview of the Semantic Web">
            <conference rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                ISWC_2002"/>
            <title>   OIL   </title>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Web_Languages"/>
            <topic rdf:resource="http://annotation.semanticweb.org/iswc/iswc.daml#
                Semantic_Annotation"/>
            <year>   2003   </year>
        </InProceedings>
        </rdf:RDF>
         -->
    </head>
    <body>
        <h2>   Languages of the Semantic Web   </h2>
        <p>OIL is a new method to encode data. The information provided allows
            inferences to be made. </p>
    </body>
</html>
```

Appendix B

Term Root Analysis of Sample Documents

Term Root Analysis			
Root	**Term**	**Stop**	**Documents**
a	a	YES	B, C, D
accomplish	accomplished		B
accord	according		B
add	add		A, C
addi	addition		B
agent	agents		C
allow	allows		D
an	an	YES	B
and	and	YES	A,B,C,D
be	be, been, being	YES	A,B,C,D
Berners-Lee	Berners-Lee		C
beyond	beyond		C
by	by	YES	B, C
can	can	YES	A
communit	communities		A,C
concept	concepts		C
data	data		A,D
design	designed		A, B
develop	developed, development		B
docu	documents		A,B
enable	enable		B,C
encode	encode		D
engin	engines		A
enhance	enhance		A,B
extend	extend		C
for	for	YES	A,B,C
has	has	YES	B
hierarchy	hierarchy		B
i	is	YES	D

include	including		B
infer	infer		C
inferenc	inferences		D

Term Root Analysis (Continued)

Root	**Term**	**Stop**	**Documents**
inform	information		A,D
is	is	YES	B,C
it	it	YES	C
keyword	keywords		C
knowledge	knowledge		B,C
language	languages		B
logic	logic		C
machine	machines		B
made	made		D
method	method		D
more	more		A
new	new		D
of	of	YES	A, B, C
OIL	OIL		D
on	on	YES	A. B. C
ontolog	ontology		B
particular	particular		B
perform	performance		A
project	project		C
provid	provide, provided		A,D
RDF	RDF		B
search	search		A
Semant	Semantic		A, B, C
standard	standards		B
subject	subject		B
support	support		B
that	that	YES	B, C
the	the	YES	A, B, C, D
these	these	YES	B
this	this	YES	B
through	through	YES	B
Tim	Tim		C
to	to	YES	A, B, C, D
understand	understanding		A

understood	understood		B
us	used, users, use	YES	A, B, C
W3C	W3C		B
Term Root Analysis (Continued)			
Root	**Term**	**Stop**	**Documents**
Web	Web		A, B, C
which	which	YES	B, C
Term Root Analysis			
will	will	YES	C
work	worked		C
XML	XML		B

Appendix C

Description of Input Files

Four input files are required for the SWAS system:

1. RDF triples for DAML version of SWIC ontology
2. RDF triples for OWL version of SWIC ontology
3. RDF triples for annotations of documents published using SWIC ontology
4. Input Parameters for SWAS system

Samples of these files are available online at http://home.cfl.rr.com/lookhome/joelynn/SWASinput.

RDF triples for DAML version of SWIC ontology: This is a spreadsheet containing three columns representing the subject, predicate and object of the RDF triples for the ontology using the DAML specification. There is one row for each statement in the ontology. An ODBC connection must be set up for this file as outlined in Appendix E, and the name of that connection should be specified as line 12 in the InputParameters.dat file.

RDF triples for OWL version of SWIC ontology: This is a spreadsheet containing three columns representing the subject, predicate and object of the RDF triples for the ontology using the OWL specification. There is one row for each statement in the ontology. An ODBC connection must be set up for this file as outlined in Appendix E, and the name of that connection should be specified as line 13 in the InputParameters.dat file.

RDF triples for annotations of documents published using SWIC ontology: This is a multi-worksheet spreadsheet representing the text and RDF triples for the document set being analyzed. The SWAS system assumes that the worksheets will have the default Microsoft Excel names, beginning with "Sheet1". Each worksheet contains 3 columns, representing the subject, predicate and object of the RDF triples. Row 1 of each spreadsheet contains the text of the document, which is an abstract of the actual paper. The RDF triples begin at row 2. An ODBC connection must be set up for this file as outlined in Appendix E, and the name of that connection should be specified as line 14 in the InputParameters.dat file

Input Parameters for SWAS system: The file "InputParameters.dat" is assumed to be in the same directory as the SWAS executable file. This is a text file with each row representing a specific input parameter as defined in the chart below.

InputParameters.dat Specification		
Line	**Data Type**	**Description**
1	Integer	Number of branches from the root of the ontology tree to be considered. In this analysis, "formal language" and "topic" are the concepts to be considered from the ontology tree root.
2	Integer	Maximum integer value to be used in finding minimum values.
3	Integer	Maximum number of RDF statements in the ontology.
4	Integer	Maximum number of unique words in any one document text. in
5	Integer	Maximum number of superconcepts for any concept in the ontology.

InputParameters.dat Specification (Continued)		
Line	**Data Type**	**Description**
6	Integer	Maximum number of lowest parents from which other concepts are derived.
7	Integer	Maximum number of valid document annotation statements.
8	Integer	Maximum number of unique term roots contained in all documents.
9	Integer	Originally, maximum number of concepts in one document. Not used in latest version of software.
10	Integer	Maximum number of documents to be analyzed.
11	Integer	Minimum number of common terms required for document pair to be included in selection set.
12	Text	Parameter to specify database connection for DAML version of ontology.
13	Text	Parameter to specify database connection for OWL version of ontology.
14	Text	Parameter to specify database connection for document text and annotations file.
15	Text	RDF specifier in predicate field to denote subconcepts in ontology.
16	Text	RDF specifier in predicate field of document annotation to denote the subject of this statement can be title of document.
17	Text	Alternate RDF specifier in predicate field of document annotation to denote the subject of this statement can be title of document.
18	Text	RDF specifier in object field of document annotation to denote the subject of this statement can be title of document.
19	Text	Alternate RDF specifier in object field of document annotation to denote the subject of this statement can be title of document.
20	Text	RDF specifier in predicate field of document annotation to denote the subject of this statement is concept of interest by rule 1.
21	Text	Alternate RDF specifier in predicate field of document annotation to denote the subject of this statement is concept of interest by rule 1.
22	Text	Second alternate RDF specifier in predicate field of document annotation to denote the subject of this statement is concept of interest by rule 1.
23	Text	RDF specifier in predicate field of document annotation to denote the subject of this statement is concept of interest by rule 2.
24	Text	Alternate RDF specifier in predicate field of document annotation to denote the subject of this statement is concept of interest by rule 2.
25	Text	Second alternate RDF specifier in predicate field of document annotation to denote the subject of this statement is concept of interest by rule 2.
26	Text	RDF specifier in object field of document annotation to denote the subject of this statement is document title.
27	Text	Alternate RDF specifier in object field of document annotation to denote the subject of this statement is document title.

InputParameters.dat Specification (Continued)		
Line	**Data Type**	**Description**
28	Text	RDF specifier in predicate field of document annotation to denote the subject of this statement is document title.
29	Text	Alternate RDF specifier in predicate field of document annotation to denote the subject of this statement is document title.
30	Text	Second alternate RDF specifier in predicate field of document annotation to denote the subject of this statement is document title.
31	Text	RDF specifier in predicate field to denote the subject of this statement is literal name of ontology concept.
32	Text	Alternate RDF specifier in predicate field to denote the subject of this statement is literal name of ontology concept.
33	Text	Second alternate RDF specifier in predicate field to denote the subject of this statement is literal name of ontology concept.
34	Text	RDF specifier in predicate field of ontology to denote the subject of this statement is concept of interest by rule 1.
35	Text	Alternate RDF specifier in predicate field of ontology to denote the subject of this statement is concept of interest by rule 1.
36	Text	RDF specifier in predicate field of ontology to denote the subject of this statement is concept of interest by rule 2.
37	Text	Alternate RDF specifier in predicate field of ontology to denote the subject of this statement is concept of interest by rule 2.
38	Integer	Number of times a document can be deselected before it is no longer considered for selection set.
39	Real	Threshold for case 1 based on doc_ratio and common term roots.
40	Real	Threshold for case 2 based on doc_ratio and common unexpected term roots.
41	Real	Threshold for case 3 based on parent-ratio and common term roots.
42	Real	Threshold for case 4 based on parent_ratio and common unexpected term roots.
43	Text	Identifier to be included in output file designation

Appendix D

Results of Correlation Comparison

Selected portions of data are shown in tables below. Complete results are available online at http://home.cfl.rr.com/lookhome/joelynn/SWASoutput.

Correlation and Ranked Correlation Rule 2 MinCommon 1 Document Ratio Common Roots (Measures 1-4))
(Significant Correlations Shown in Bold)

Thres-hold	#Prs Sel	Min rho	Max Val	MaxMeas	M1	M2	M3	M4
0.128000	28	0.317	0.435	JMatch	-0.280	-0.215	-0.627	-0.582
	Rank				16	15	18	17
	Pts				0	0	0	0
0.155000	561	0.070	0.340	RankJMatch	-0.111	0.012	-0.116	-0.018
	Rank				17	15	18	16
	Pts				0	0	0	0
0.175000	1176	0.048	0.242	RankJMatch	-0.167	-0.086	-0.160	-0.108
	Rank				18	15	17	16
	Pts				0	0	0	0
0.217750	1770	0.039	0.163	RankJMatch	-0.198	-0.153	-0.226	-0.200
	Rank				16	15	18	17
	Pts				0	0	0	0
1.0000	2346	0.034	0.114	RankJMatch	-0.088	-0.042	-0.118	-0.098
	Rank				16	15	18	17
	Pts				0	0	0	0
	Total Pts			RankJMatch	0	0	0	0
	Rank				15	15	15	15

Correlation and Ranked Correlation Rule 2 MinCommon 1 Document Ratio Common Roots (Measures 5-8)

(Significant Correlations Shown in Bold)

Thres-hold	#Prs Sel	Min rho	Max Val	MaxMeas	M5	M6	M7	M8
0.128000	28	0.317	0.435	JMatch	**0.435**	**0.405**	**0.386**	**0.369**
	Rank				1	4	8	9
	Pts				18	15	11	10
0.155000	561	0.070	0.340	RankJMatch	**0.264**	**0.340**	**0.202**	**0.241**
	Rank				4	1	9	5
	Pts				15	18	10	14
0.175000	1176	0.048	0.242	RankJMatch	**0.183**	**0.242**	**0.141**	**0.148**
	Rank				7	1	11	9
	Pts				12	18	8	10
0.217750	1770	0.039	0.163	RankJMatch	**0.137**	**0.163**	**0.131**	**0.102**
	Rank				6	1	7	11
	Pts				13	18	12	8
1.0000	2346	0.034	0.114	RankJMatch	**0.069**	**0.114**	**0.059**	**0.041**
	Rank				5	1	7	10
	Pts				14	18	12	9
	Total Pts			RankJMatch	72	87	53	51
	Rank				4	1	6	7

Correlation and Ranked Correlation Rule 2 MinCommon 1 Document Ratio Common Roots (Measures 9-12)

(Significant Correlations Shown in Bold)

Thres-hold	#Prs Sel	Min rho	Max Val	MaxMeas	M9	M10	M11	M12
0.128000	28	0.317	0.435	JMatch	**0.435**	**0.404**	**0.398**	**0.389**
	Rank				2	5	6	7
	Pts				17	14	13	12
0.155000	561	0.070	0.340	RankJMatch	**0.270**	**0.321**	**0.133**	**0.184**
	Rank				3	2	14	12
	Pts				16	17	5	7
0.175000	1176	0.048	0.242	RankJMatch	**0.201**	**0.227**	**0.068**	**0.086**
	Rank				4	2	14	13
	Pts				15	17	5	6
0.217750	1770	0.039	0.163	RankJMatch	**0.142**	**0.147**	**0.078**	**0.073**
	Rank				4	3	12	13
	Pts				15	16	7	6
1.0000	2346	0.034	0.114	RankJMatch	**0.074**	**0.098**	**0.051**	**0.051**
	Rank				4	3	9	8
	Pts				15	16	10	11
	Total Pts			RankJMatch	78	80	40	42
	Rank				3	2	11	9

Correlation and Ranked Correlation Rule 2 MinCommon 1 Document Ratio Common Roots (Measures 13-16)

(Significant Correlations Shown in Bold)

Thres-hold	#Prs Sel	Min rho	Max Val	MaxMeas	M13	M14	M15	M16
0.128000	28	0.317	0.435	Jmatch	0.274	0.256	0.221	**0.355**
	Rank				12	13	14	10
	Pts				0	0	0	9
0.155000	561	0.070	0.340	RankJMatch	**0.208**	**0.218**	**0.144**	**0.211**
	Rank				8	6	13	7
	Pts				11	13	6	12
0.175000	1176	0.048	0.242	RankJMatch	**0.209**	**0.192**	**0.148**	**0.189**
	Rank				3	5	10	6
	Pts				16	14	9	13
0.217750	1770	0.039	0.163	RankJMatch	**0.141**	**0.116**	**0.131**	**0.156**
	Rank				5	9	8	2
	Pts				14	10	11	17
1.0000	2346	0.034	0.114	RankJMatch	0.016	-0.010	**0.061**	**0.101**
	Rank				12	13	6	2
	Pts				0	0	13	17
	Total Pts			RankJMatch	41	37	39	68
	Rank				10	13	12	5

Correlation and Ranked Correlation Rule 2 MinCommon 1
Document Ratio Common Roots (Measures 17-18)
(Significant Correlations Shown in Bold)

Thres-hold	#Prs Sel	Min rho	Max Val	MaxMeas	M17	M18
0.128000	28	0.317	0.435	JMatch	**0.418**	0.355
	Rank				3	11
	Pts				16	8
0.155000	561	0.070	0.340	RankJMatch	**0.185**	**0.191**
	Rank				11	10
	Pts				8	9
0.175000	1176	0.048	0.242	RankJMatch	**0.156**	**0.119**
	Rank				8	12
	Pts				11	7
0.217750	1770	0.039	0.163	RankJMatch	**0.102**	**0.056**
	Rank				10	14
	Pts				9	5
1.0000	2346	0.034	0.114	RankJMatch	0.023	-0.014
	Rank				11	14
	Pts				0	0
	Total Pts			RankJMatch	44	29
	Rank				8	14

Points for All Cases Simple Match Portion								
Rule	MinCom	Case	MaxVal	MaxMeas	Simple Match	Ranked Simple Match	Simple Match Inferred	Ranked Simple Match Inferred
2	1	DocCR	87	RankJMatch	0	0	0	0
2	1	DocCUR	69	RankJMatch	0	18	0	15
2	1	ParCR	80	RankJMatch	0	0	0	0
2	1	ParCUR	69	RankJMatch	15	56	13	27
2	0	DocCR	85	RankJMatch	0	0	0	0
2	0	DocCUR	84	RankJMatch	0	18	0	15
2	0	ParCR	80	RankJMatch	0	0	0	0
2	0	ParCUR	69	RankJMatch	15	56	13	27
1	1	DocCR	84	RankJMatch	0	0	0	0
1	1	DocCUR	88	RankJMatch	0	0	0	0
1	0	DocCR	88	RankJMatch	0	0	0	0
1	0	DocCUR	88	RankJMatch	0	0	0	0
3	1	DocCR	89	RankJMatch	0	3	0	0
3	1	DocCUR	85	RankJMatch	0	4	0	6
3	1	ParCR	89	RankJMatch	11	10	14	0
3	1	ParCUR	89	RankJMatch	11	10	14	0
3	0	DocCR	82	RankJMatch	0	0	0	0
3	0	DocCUR	86	JMatch	0	0	0	0
3	0	ParCR	78	RankJMatch	0	0	0	0
3	0	ParCUR	89	RankJMatch	11	10	14	0
		Total	**1649**	**RankJMatch**	**63**	**185**	**68**	**90**
		Rank			**18**	**15**	**17**	**16**

Points for All Cases – Jacquard Match Portion								
Rule	MinCom	Case	MaxVal	MaxMeas	Jacqurd Match	Ranked Jacquard Match	Jacquared Match Inferred	Ranked Jacquared Match Inferred
2	1	DocCR	87	RankJMatch	72	87	53	51
2	1	DocCUR	69	RankJMatch	53	69	33	17
2	1	ParCR	80	RankJMatch	60	80	64	59
2	1	ParCUR	69	RankJMatch	46	69	20	20
2	0	DocCR	85	RankJMatch	54	85	59	56
2	0	DocCUR	84	RankJMatch	53	84	33	33
2	0	ParCR	80	RankJMatch	60	80	64	59
2	0	ParCUR	69	RankJMatch	46	69	20	20
1	1	DocCR	84	RankJMatch	66	84	66	84
1	1	DocCUR	88	RankJMatch	62	88	62	88
1	0	DocCR	88	RankJMatch	70	88	70	88
1	0	DocCUR	88	RankJMatch	62	88	62	88
3	1	DocCR	89	RankJMatch	69	89	43	39
3	1	DocCUR	85	RankJMatch	84	85	62	27
3	1	ParCR	89	RankJMatch	80	89	38	23
3	1	ParCUR	89	RankJMatch	80	89	38	23
3	0	DocCR	82	RankJMatch	73	82	59	54
3	0	DocCUR	86	Jmatch	86	77	46	29
3	0	ParCR	78	RankJMatch	65	78	68	62
3	0	ParCUR	89	RankJMatch	80	89	38	23
		Total	**1649**	**RankJMatch**	**1321**	**1649**	**998**	**943**
		Rank			**3**	**1**	**6**	**7**

Points for All Cases – Cosine Section								
Rule	MinCom	Case	MaxVal	MaxMeas	Cosine	Ranked Cosine	Cosine Inferred	Ranked Cosine Inferred
2	1	DocCR	87	RankJMatch	78	80	40	42
2	1	DocCUR	69	RankJMatch	47	62	0	0
2	1	ParCR	80	RankJMatch	56	72	57	70
2	1	ParCUR	69	RankJMatch	40	61	24	27
2	0	DocCR	85	RankJMatch	72	78	43	49
2	0	DocCUR	84	RankJMatch	61	80	0	0
2	0	ParCR	80	RankJMatch	56	72	57	70
2	0	ParCUR	69	RankJMatch	40	61	24	27
1	1	DocCR	84	RankJMatch	55	74	55	74
1	1	DocCUR	88	RankJMatch	41	80	41	80
1	0	DocCR	88	RankJMatch	60	66	60	66
1	0	DocCUR	88	RankJMatch	41	80	41	80
3	1	DocCR	89	RankJMatch	78	85	32	30
3	1	DocCUR	85	RankJMatch	80	78	11	15
3	1	ParCR	89	RankJMatch	61	82	10	11
3	1	ParCUR	89	RankJMatch	61	82	10	11
3	0	DocCR	82	RankJMatch	80	80	40	39
3	0	DocCUR	86	Jmatch	81	74	6	9
3	0	ParCR	78	RankJMatch	65	75	53	55
3	0	ParCUR	89	RankJMatch	61	82	10	11
		Total	**1649**	**RankJMatch**	**1214**	**1504**	**614**	**766**
		Rank			**4**	**2**	**11**	**9**

Points for All Cases- GCSM section						
Rule	MinCom	Case	MaxVal	MaxMeas	Gcsm	Ranked Gcsm
2	1	DocCR	87	RankJMatch	41	37
2	1	DocCUR	69	RankJMatch	40	37
2	1	ParCR	80	RankJMatch	6	0
2	1	ParCUR	69	RankJMatch	0	0
2	0	DocCR	85	RankJMatch	41	37
2	0	DocCUR	84	RankJMatch	40	54
2	0	ParCR	80	RankJMatch	6	0
2	0	ParCUR	69	RankJMatch	0	0
1	1	DocCR	84	RankJMatch	0	0
1	1	DocCUR	88	RankJMatch	0	0
1	0	DocCR	88	RankJMatch	0	0
1	0	DocCUR	88	RankJMatch	0	0
3	1	DocCR	89	RankJMatch	28	23
3	1	DocCUR	85	RankJMatch	17	37
3	1	ParCR	89	RankJMatch	0	0
3	1	ParCUR	89	RankJMatch	0	0
3	0	DocCR	82	RankJMatch	35	26
3	0	DocCUR	86	Jmatch	28	30
3	0	ParCR	78	RankJMatch	6	5
3	0	ParCUR	89	RankJMatch	0	0
		Total	**1649**	**RankJMatch**	**288**	**286**
		Rank			**13**	**14**

Points for All Cases – UHI and WHI Section								
Rule	MinCom	Case	MaxVal	MaxMeas	UHI	Ranked UHI	WHI	Ranked WHI
2	1	DocCR	87	RankJMatch	39	68	44	29
2	1	DocCUR	69	RankJMatch	19	39	16	17
2	1	ParCR	80	RankJMatch	63	79	39	36
2	1	ParCUR	69	RankJMatch	25	44	0	0
2	0	DocCR	85	RankJMatch	39	73	27	21
2	0	DocCUR	84	RankJMatch	19	39	16	17
2	0	ParCR	80	RankJMatch	63	79	39	36
2	0	ParCUR	69	RankJMatch	25	44	0	0
1	1	DocCR	84	RankJMatch	55	74	53	74
1	1	DocCUR	88	RankJMatch	41	80	42	80
1	0	DocCR	88	RankJMatch	60	66	57	66
1	0	DocCUR	88	RankJMatch	41	80	42	80
3	1	DocCR	89	RankJMatch	20	40	54	62
3	1	DocCUR	85	RankJMatch	47	58	25	20
3	1	ParCR	89	RankJMatch	37	43	7	8
3	1	ParCUR	89	RankJMatch	37	43	7	8
3	0	DocCR	82	RankJMatch	57	71	57	34
3	0	DocCUR	86	JMatch	42	60	29	23
3	0	ParCR	78	RankJMatch	68	69	52	33
3	0	ParCUR	89	RankJMatch	37	43	7	8
		Total	**1649**	**RankJMatch**	**834**	**1192**	**613**	**652**
		Rank			**8**	**5**	**12**	**10**

Points for Rule 2 Aggregated for Each Measure								
Rule	MinCom	Case	SimpleM	Jacquard	Cosine	GCSM	UHI	WHI
2	1	DocCR	0	263	240	156	214	146
2	1	DocCUR	33	172	109	154	116	66
2	1	ParCR	0	263	255	12	284	150
2	1	ParCUR	111	155	152	0	138	0
2	0	DocCR	0	254	242	156	224	96
2	0	DocCUR	33	203	141	188	116	66
2	0	ParCR	0	263	255	12	284	150
2	0	ParCUR	111	155	152	0	138	0
		Total	**288**	**1728**	**1546**	**678**	**1514**	**674**
		Rank	**6**	**1**	**2**	**4**	**3**	**5**

Points for Rule 2 Aggregated for Each Measure DocRatio Selection

Rule	MinCom	Case	SimpleM	Jacquard	Cosine	GCSM	UHI	WHI
2	1	DocCR	0	263	240	156	214	146
2	1	DocCUR	33	172	109	154	116	66
2	0	DocCR	0	254	242	156	224	96
2	0	DocCUR	33	203	141	188	116	66
		Total	**66**	**892**	**732**	**654**	**670**	**374**
		Rank	**6**	**1**	**2**	**4**	**3**	**5**

Points for Rule 2 Aggregated for Each Measure ParentRatio Selection								
Rule	MinCom	Case	SimpleM	Jacquard	Cosine	GCSM	UHI	WHI
2	1	ParCR	0	263	255	12	284	150
2	1	ParCUR	111	155	152	0	138	0
2	0	ParCR	0	263	255	12	284	150
2	0	ParCUR	111	155	152	0	138	0
		Total	**222**	**836**	**814**	**24**	**844**	**300**
		Rank	**5**	**2**	**3**	**6**	**1**	**4**

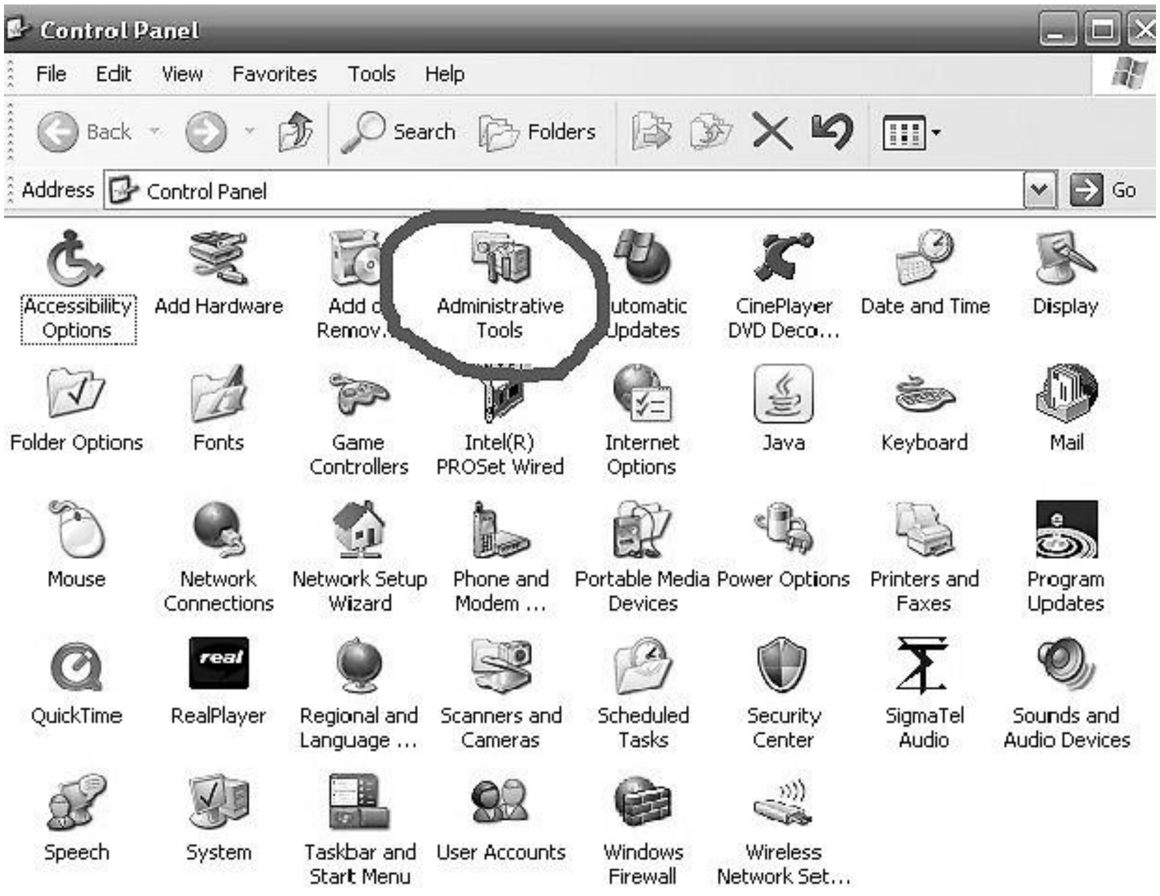
Appendix E

ODBC Setup

The following example demonstrates the steps to set up an object control database for a Microsoft Excel file on a computer utilizing a Windows XP Operating System. The exact setup steps will vary depending on the operating system.

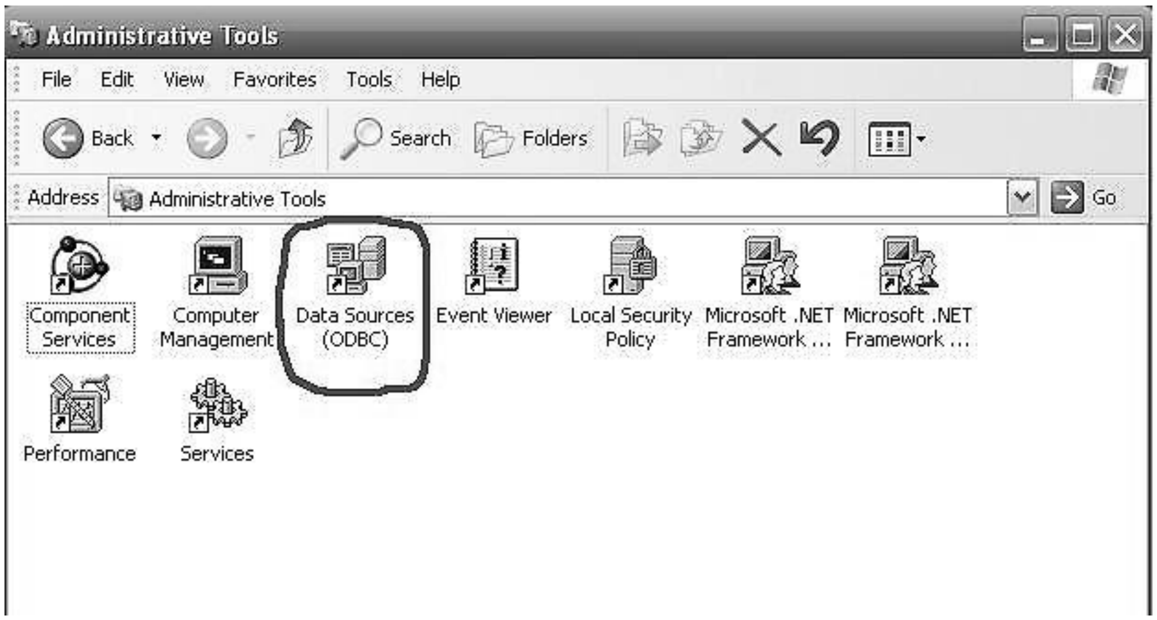
From the Control Panel:

Choose Administrative Tools



From Administrative Tools screen:

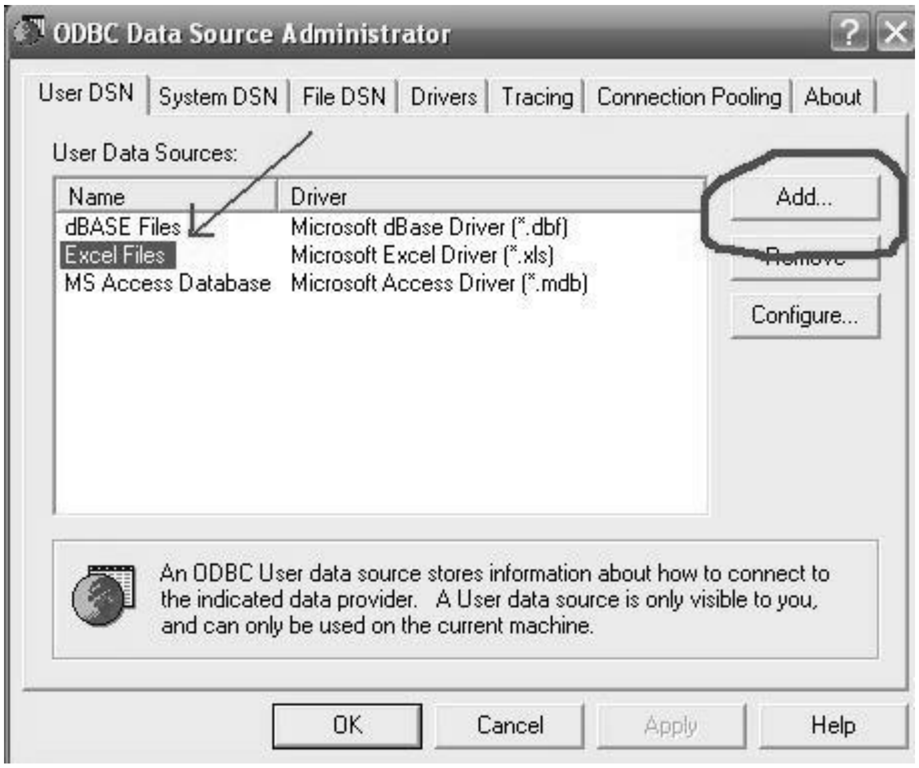
Choose Data Sources (ODBC)



From Data Sources (ODBC):

Choose Excel Files

Choose Add



User Data Sources:

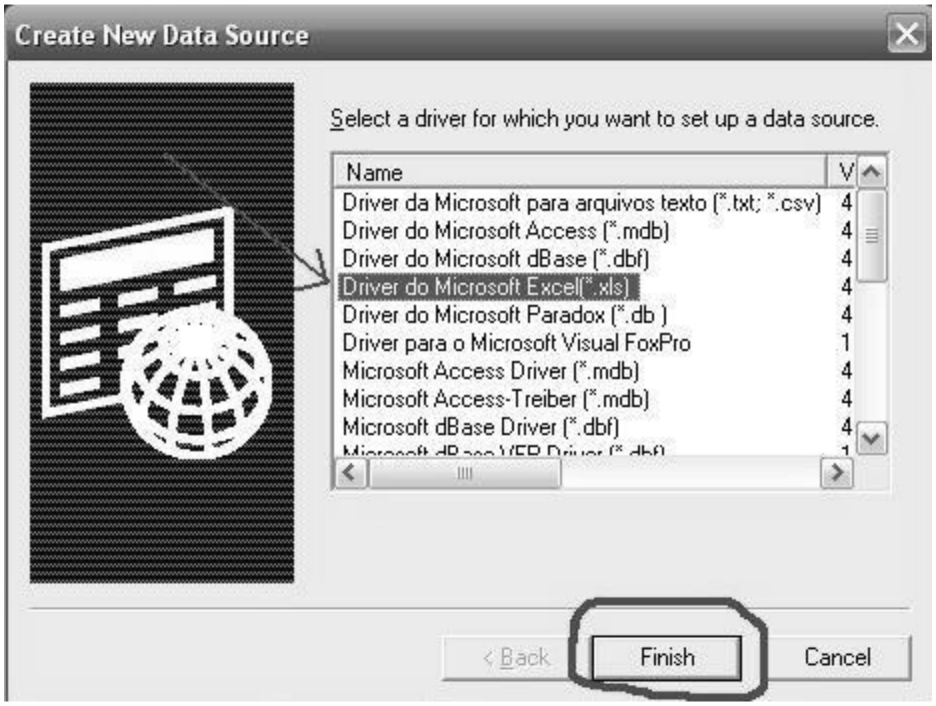
Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
MS Access Database	Microsoft Access Driver (*.mdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

From Create New Data Source:

Choose Driver do Microsoft Excel

Click Finish



Select a driver for which you want to set up a data source.

Name	V
Driver da Microsoft para arquivos texto (*.txt; *.csv)	4
Driver do Microsoft Access (*.mdb)	4
Driver do Microsoft dBase (*.dbf)	4
Driver do Microsoft Excel(*.xls)	4
Driver do Microsoft Paradox (*.db)	4
Driver para o Microsoft Visual FoxPro	1
Microsoft Access Driver (*.mdb)	4
Microsoft Access-Treiber (*.mdb)	4
Microsoft dBase Driver (*.dbf)	4
Microsoft dBase VFP Driver (*.dbf)	1

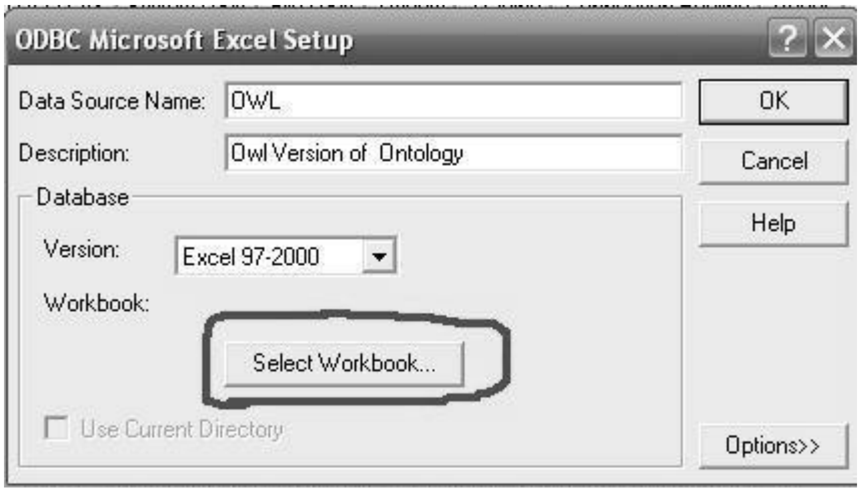
In the “Datasource Name” box:

Specify name as entered in Input Parameters file

In the “Description” box:

Enter description

Click “Select Workbook”



Data Source Name: OWL

Description: Owl Version of Ontology

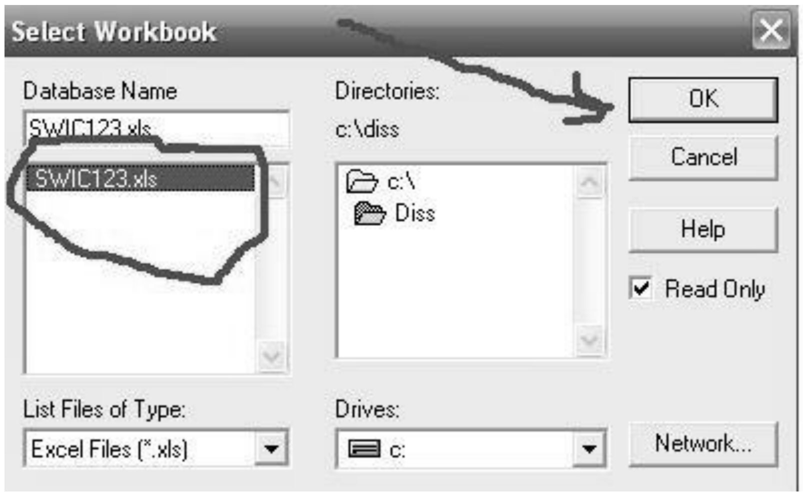
Version: Excel 97-2000

From "Select Workbook" screen:

Navigate to folder where file is located

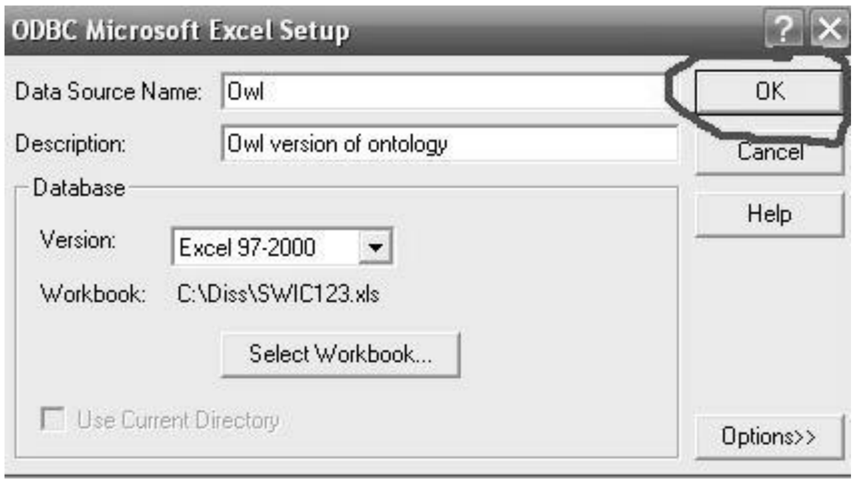
Select correct file

Click OK.



From ODBC Microsoft Excel Setup screen:

Click OK



Data Source Name: Owl

Description: Owl version of ontology

Database

Version: Excel 97-2000

Workbook: C:\Diss\SWIC123.xls

Use Current Directory

The ODBC connection can now be integrated into required software. When using with java programs, a JCBC/ODBC connection is required through the use of the Connection class.

Appendix F

Source Code of the SWAS System

```
import java.io.*;

public class SWAS {

//This is the primary class for the SWAS system  (utilize default constructor)

public static void main (String args[]){

//triggers execution of methods in other classes

    InputParameters ip = new InputParameters();

    System.out.println("Begin SWAS...read input for this run");

    ip.readInput( );

    System.out.println("RunID is " + ip.runID+" and docUsed is " + ip.docUsed);

    System.out.println("Process ontology");

    RDFConceptArray araConcept = new RDFConceptArray(ip);

    Ontology onto = new Ontology(araConcept,ip);

    onto.readRDF();

    onto.findConcepts();

    onto.findLevels();

    onto.findSuperConcepts();

    onto.findConceptsLists( );

    for (int i=0; i<onto.conceptCt; i++){
```

```
    if (onto.concepts.numSuperConcepts[i] == 0) {
        System.out.println(" Concept " + i + " " + onto.concepts.name[i] +
                                                    " has no superConcepts:");
    }
    else {
        System.out.println(" Concept " + i + " " + onto.concepts.name[i] +
                                                    " has superConcepts:");
        for (int j=0; j<onto.concepts.numSuperConcepts[i]; j++){
            System.out.print("\t"+onto.concepts.superList[i][j]);
        }
        System.out.println( );
    }
}
Ancestors ancestors = new Ancestors(onto.conceptCt);
ancestors.findCAs(onto.concepts, onto.conceptCt, ip.maxVal);
ancestors.findVectors(onto.concepts, onto.conceptCt);
System.out.println("Calculate ontology term similarity");
onto.calcTermSim(ancestors);
System.out.println("Process annotations and text in documents");
Annotations annot = new Annotations(onto,ip, ancestors.numParents);
annot.readAnnotStatements(ancestors, onto, araConcept, ip);
Stop stop = new Stop();
Words wt = new Words();
```

```
TermDocumentMatrix tdm = new TermDocumentMatrix (ip.docUsed, ip.totWords);
for (int nd=0; nd<ip.docUsed;nd++){
    DocText dt = new DocText(annot.text[nd]);
    Roots textRoot = new Roots();
    wt = dt.parseWords();
    wt.stripWords( );
     wt.sortWords(0,wt.wordCt-1);
    wt.elimDups();
    stop.elimStops(wt);
    wt.findRoots(textRoot);
    wt.sortWords(0,wt.wordCt-1);
    wt.elimDups();
    tdm.addDoc(nd,wt);
}
tdm.createMaster(annot, ip.docUsed);
System.out.println("Find roots in " + ip.docUsed+ " documents");
tdm.findRoots(ip.runID, annot, ip.docUsed );
tdm.assignStatus(annot, onto, ip.docUsed);
annot.calcRelativity(onto, tdm, ip);
System.out.println("Calculate statistics for "+ ip.docUsed+ " documents" );
Stats stats = new Stats (annot, ip);
annot.calcCommon(tdm,stats,ip.docUsed);
stats.calcSimilarity(annot, ancestors, onto, ip.runID, ip.docUsed, ip.selectThresh, tdm,
                                              ip.numDeSels, ip.minCommon);
    System.out.println("\nEnd of program");
}
}



class Ancestors{
//This class defines the heirarchical relationship of concepts
double [ ] [ ] lVector;
int  [ ] [ ] commonAncestorArray;
int [ ] conceptToParent;
int [ ] hiParentArray;
int [ ] parentArray;
int numParents;
boolean hier;

Ancestors ( ) {
    System.err.println("***Warning Default Ancestors Constructed with no max number
of
                                            docs");
    System.err.println("***Program will exit");
    System.exit(1);
}
```

```
Ancestors (int size) {
    commonAncestorArray = new int [size] [size];
    hiParentArray = new int [size];
    conceptToParent= new int[size];
    parentArray = new int [size];
    lVector = new double [size] [size];
    for (int rc=0;rc<size;rc++){
        for (int cc=0;cc<size;cc++){
            commonAncestorArray[rc][cc] = -1;
            if (rc== cc){
                lVector [rc] [cc] = 1;
            }
            else {
                lVector [rc] [cc] = 0;
            }
    }
    hiParentArray[rc] = -1;
    conceptToParent[rc] = -1;
    parentArray[rc] = -1;
    }
}
```

```
void findCAs( RDFConceptArray concepts, int numConcepts, int ancMax){
//finds level of least common ancestor and concept index of the first
//common ancestor (closest to root of concept tree) for each concept of interest
    int lca=-1;
    int hca=-1;
    for (int nr = 0; nr<numConcepts; nr++){
        for (int   nc = 0; nc< nr; nc++){
            lca = findCommonAncestor(nr, nc, concepts, numConcepts);
            commonAncestorArray[nr][nc] = lca;
            commonAncestorArray[nc][nr] = lca;
        }
        hca = findLoParent(nr, concepts, numConcepts, ancMax);
        hiParentArray[nr] = hca;
    }
    numParents = createParentArray(numConcepts, ancMax);
    hier=(numConcepts>numParents);
    relateParents(numConcepts);
}


void findVectors ( RDFConceptArray concepts, int numConcepts){
//find the dot product of elaf nodes for the Gcsm measure
    for (int nr = 0; nr<numConcepts; nr++){
        lVector[nr][nr]=1;
        for (int   nc = nr+1; nc< numConcepts; nc++){
            double lVect = (double)(2*commonAncestorArray[nr][nc]) /
                                                    (double)(concepts.lowestLevel[nr] +
concepts.lowestLevel[nc]);
            lVector[nr] [nc] = lVect;
            lVector[nc] [nr] = lVect;
        }
    }
}

int findCommonAncestor(int nr, int nc, RDFConceptArray concepts, int numconcepts) {
//determines the level of the common ancestor closest to root of tree
//and stores that value
    boolean common = true;
    boolean found=false;
    String checkconcept;
    int maxSoFar=0;
    int thisPair;
    if ((concepts.numSuperConcepts[nr]==0) || (concepts.numSuperConcepts[nc]==0))
        common=false;
    if (common){
        for (int tr=0;tr<concepts.numSuperConcepts[nr];tr++){
            found = false;
            checkconcept = concepts.superList[nr][tr];
            for (int tc=0;tc< concepts.numSuperConcepts[nc];tc++){
                if (checkconcept.equals(concepts.superList[nc][tc])){
                    thisPair=concepts.lookUpLevel(checkconcept,numconcepts);
                    if (thisPair>maxSoFar)
                        maxSoFar=thisPair;
                }
            }
        }
    }
    if (maxSoFar==0)
        maxSoFar=checkBranch(nr,nc,concepts);
    return maxSoFar;
}

int findLoParent(int nr, RDFConceptArray concepts, int numconcepts, int ancMax) {
//finds the parent node closest to root of hierarchical tree
    boolean found=false;
    String checkconcept;
    int minSoFar=ancMax;
    int minLevel=ancMax;
    int thisLevel;
    if (concepts.numSuperConcepts[nr]!=0){
        for (int tr=0;tr<concepts.numSuperConcepts[nr];tr++){
            found = false;
            checkconcept = concepts.superList[nr][tr];
            thisLevel=concepts.lookUpLevel(checkconcept,numconcepts);
            if (thisLevel<minLevel){
                minLevel=thisLevel;
                minSoFar=concepts.lookUpIndex(checkconcept,numconcepts);
            }
        }
    }
    else{//no superConcepts, this is parent
        minSoFar = nr;
    }
    if (minSoFar==ancMax){ // no parent found, this node is parent
        minSoFar=nr;
    }
    return minSoFar;
}

int createParentArray (int num, int ancMax){
//assigns parent value to each concept
    int count=0;
    int min;
    int last = -1;
    for (int pa=0; pa<num; pa++){
        min = ancMax;
        for (int n=0; n<num; n++){
            if ((hiParentArray[n]<min) && (hiParentArray[n] > last)){
                min=hiParentArray[n];
            }
        }
        if (min<ancMax){
            parentArray[pa] = min;
            last = min;
            count++;
        }
    }
    return count;
}

void relateParents (int num){
//creates index to relate concepts to parent in parent array
    int val;
    int index;
    for (int nc=0; nc<num; nc++){
        val = hiParentArray[nc];//
        index = -1;
        int np = 0;
        while (np<numParents && index < 0) {
            if (val == parentArray[np]){
                index=np;
            }
            np++;
        }
        conceptToParent[nc] = index;
    }
}


int checkBranch(int nr,int nc, RDFConceptArray concepts){
//determines if concepts are derived from two main rules to identify
//concepts of interest
    if (concepts.branch[nr]== concepts.branch[nc])
        return 1;
    else
        return 0;
    }
}
```

```
import java.sql.*;

import java.io.*;

import sun.jdbc.odbc.*;

import java.text.*;

class Annotations  {

//This class defines the properties of annotations found in documents

boolean [ ] selectedSheet;

boolean [ ] validSheet;

double [ ][ ] pWeight;

double [ ][ ] pInfWeight;

int [ ] countAnnotsInDoc;

int [ ] countInfersInDoc;

int [ ] [ ] countParentInfMatrix;

int [ ] [ ] countParentMatrix;

int [ ] [ ] docAnnotMatrix;

int [ ] [ ] docParentInfMatrix;

int [ ] [ ] docParentMatrix;

int [ ] docRemoved;

int [ ] numStmtsInDoc;

int numValidDocs;

String [ ] [ ] annotDocsObj;

String [ ] [ ] annotDocsPred;

String [ ] [ ] annotDocsSub;
String [ ] docTitles;
String [ ] text;

Annotations( ){
    System.err.println("***Warning Default Annotations Constructed with 0
documents");
    System.err.println("***Program will exit");
    System.exit(1);
}

public Annotations (Ontology onto, InputParameters ip, int numParents) {
text = new String [ip.docUsed];
    pWeight = new double[ip.docUsed] [numParents];
    pInfWeight = new double[ip.docUsed] [numParents];
    annotDocsSub = new String[ip.docUsed] [ip.maxStmts];
    annotDocsPred = new String[ip.docUsed] [ip.maxStmts];
    annotDocsObj = new String[ip.docUsed] [ip.maxStmts];
    docAnnotMatrix = new int [ip.docUsed] [onto.conceptCt];
    countParentMatrix = new int[ip.docUsed][numParents];
    countParentInfMatrix = new int[ip.docUsed][numParents];
    docParentMatrix = new int[ip.docUsed] [numParents];
    docParentInfMatrix = new int[ip.docUsed] [numParents];
    countAnnotsInDoc = new int[ip.docUsed];
    countInfersInDoc = new int[ip.docUsed];
    docRemoved = new int[ip.docUsed];
    docTitles = new String[ip.docUsed];
    validSheet = new boolean[ip.docUsed];
    selectedSheet=new boolean[ip.docUsed];
    numStmtsInDoc = new int[ip.docUsed];
    String url= ip.urlUsed;
    String user = "";
    String password = "";
    initMatrix(ip.docUsed, onto.conceptCt, numParents);
}

void initMatrix(int size, int numConcepts, int numParents){
//declares arrays used to store annoations
    for (int rc = 0; rc < size; rc++){
        for (int cc = 0; cc < numConcepts; cc++) {
            docAnnotMatrix[rc][cc] = 0;
        }
        countAnnotsInDoc[rc] = 0;
        countInfersInDoc[rc] = 0;
        docRemoved[rc] = 0;
        validSheet[rc] = false;
        selectedSheet[rc] = false;
    }
}



public void readAnnotStatements(Ancestors ancestors, Ontology onto,
                                                            RDFConceptArray
araConcept, InputParameters ip){
//reads multi-worksheet spreadsheet containing the text and annotations
//from documents and stores that data
    int validNum=0;
    String subject, object, predicate;
    Connection con=null;
    String url = ip.urlUsed;
    String user="";
    String password="";
    try{
        new JdbcOdbcDriver();
        con = DriverManager.getConnection(url,user,password);
    }
    catch(Exception ex) {
        System.err.print("JDBC Connection not established Exception: ");
        System.err.println(ex.getMessage());
        System.err.println("url = " + url + " user = " + user + " password = " + password);
        ex.printStackTrace();
        System.exit(1);
    }
    String query;
    ResultSet rs=null;
    Statement st=null;
    try{
        st = con.createStatement();
    }
    catch(Exception ex) {
        System.err.print("JDBC Connection Exception: ");
        System.err.println(ex.getMessage());
        ex.printStackTrace();
        System.exit(2);
    }
    RDFStmts rdfStmts = new RDFStmts(ip.maxStmts);
    String title;
    boolean storeDoc;
    int parentNdx;
    for (int sheetNum = 1; sheetNum <= ip.docUsed; sheetNum++){
        clearRuleArray(rdfStmts, ip);
        query = "SELECT * FROM [Sheet" + Integer.toString(sheetNum)+"$]";
        try{
```

```
    rs = st.executeQuery(query);
}
catch(Exception ex) {
    System.err.print("Query execution Exception: ");
    System.err.println(ex.getMessage());
    ex.printStackTrace();
    System.exit(2);
}
int numberStatements = 0;
try{
    if (rs.next())
        text[sheetNum-1] = rs.getString(1);
    }
    catch(Exception ex) {
        System.err.print("Query cursor Exception: ");
        System.err.println(ex.getMessage());
        ex.printStackTrace();
        System.exit(2);
    }
    try {
        while (rs.next()){
            try {
                subject = rs.getString(1);
```

```
                    }
                    catch(Exception ex) {
                        System.err.println("error Sheetnum=" + sheetNum);
                        System.err.println("error numberStatements = " + numberStatements);
                        System.err.print("Attempt to read null string INPUT ERROR
Exception: ");
                        System.err.println(ex.getMessage());
                        ex.printStackTrace();
                        subject="";
                    }
                    predicate = rs.getString(2);
                    object = rs.getString(3);
                    rdfStmts.docSub[numberStatements] = subject;
                    rdfStmts.docPrd[numberStatements] = predicate;
                    rdfStmts.docObj[numberStatements] = object;
                    numberStatements++;
                }
            }
            catch(Exception ex) {
            System.err.print("After subject has been read Exception: ");
            System.err.println(ex.getMessage());
            ex.printStackTrace();
            System.exit(2);
```

```
    }
    rdfStmts.numStmts = numberStatements-1;
    if (rdfStmts.numStmts>0)
        title = findURI(rdfStmts, sheetNum-1, ip);
    else
        title = "No stmts this document docNum="+(sheetNum-1);
    int parentConceptNum=-1;
    int annotConceptNum;
    int numRules = checkRules(rdfStmts,title,ip,onto);
    boolean validDoc= numRules>0;
    if (validDoc){
        validNum++;
        for (int stmt=0;stmt<rdfStmts.numStmts;stmt++){
            if (rdfStmts.meetsRule[stmt]) {
                annotConceptNum=findConcept(rdfStmts.docObj[stmt],sheetNum,onto);
                docAnnotMatrix[sheetNum-1][annotConceptNum]=1;
                parentConceptNum =

ancestors.conceptToParent[ancestors.hiParentArray[annotConceptNum]];
                docParentMatrix[sheetNum-1][parentConceptNum] = 1;
                docParentInfMatrix[sheetNum-1][parentConceptNum] = 1;
            }
        }
        int countAnnot =0;
        int countInfer = 0;
        for (int aConcept=0;aConcept<onto.conceptCt;aConcept++){
            if (docAnnotMatrix[sheetNum-1][aConcept]==1){
                countAnnot++;
                for (int sc=0; sc<onto.conceptCt;sc++){
                    if (araConcept.superMatrix[aConcept] [sc] == 1 ){
                        if (docAnnotMatrix[sheetNum-1][sc] == 0) {
                            docAnnotMatrix[sheetNum-1][sc] = 2;
                            parentConceptNum =

ancestors.conceptToParent[ancestors.hiParentArray[sc]];
                            docParentInfMatrix[sheetNum-1][parentConceptNum] = 2;
                            countInfer++;
                        }
                    }
                }
            }
        }
        storeDoc(sheetNum-1,rdfStmts,title);
        calcParentMatrix(sheetNum-1, onto.conceptCt, ancestors);
        validSheet[sheetNum-1] = validDoc;
    }
    }
    numValidDocs = validNum;
    calcParentWeights(ancestors);
    }

void calcParentWeights(Ancestors ancestors ){
//calculates weight of parents based on how many annotations
//for this document are based on that parent as compared to
//total number of annotations for this document
    int total[ ] = new int [numValidDocs];
    int totalInf[ ] = new int[numValidDocs];
    for (int doc=0;doc<numValidDocs;doc++){
        total[doc]=0;
        totalInf[doc]=0;
        for (int parent=0; parent<ancestors.numParents; parent++){
            total[doc] = total[doc]+countParentMatrix[doc][parent];
            totalInf[doc] = totalInf[doc]+countParentInfMatrix[doc][parent];
        }
        for (int parent=0; parent<ancestors.numParents; parent++){
            pWeight[doc][parent] = (double)(countParentMatrix[doc][parent] /
                                                        (double) total[doc]);
            pInfWeight[doc][parent] = (double)(countParentInfMatrix[doc][parent] /
                                                        (double) totalInf[doc]);
        }
    }
}

void clearRuleArray(RDFStmts rdfStmts, InputParameters ip){
//initializes meets.Rule element to false for all statements
    for (int stmtNo=0; stmtNo<ip.maxStmts; stmtNo++){
        rdfStmts.meetsRule[stmtNo]=false;
    }
}

void storeDoc(int docNo, RDFStmts rdfStmts, String title){
//stores REF triple and title for all statements
    for (int st=0;st<rdfStmts.numStmts;st++) {
        annotDocsSub[docNo] [st] = rdfStmts.docSub[st];
        annotDocsPred[docNo] [st] = rdfStmts.docPrd[st];
        annotDocsObj[docNo] [st] = rdfStmts.docObj[st];
        docTitles[docNo] = title;
    }
}

int checkRules (RDFStmts rdfStmts, String title,InputParameters ip, Ontology onto){
//checks if RDF statement matches rules of interest
```

```
    int meetsCk = 0;
    for (int stmtNo=0; stmtNo<rdfStmts.numStmts; stmtNo++){
        if (rdfStmts.docSub[stmtNo] != null){
            if ((rdfStmts.docSub[stmtNo].equals(title))){
                if  (checkPred(rdfStmts.docPrd[stmtNo],ip)
                                && checkConcept(rdfStmts.docObj[stmtNo],onto)){
                    meetsCk++;
                    rdfStmts.meetsRule[stmtNo]=true;
                }
            }
        }
        else{
            rdfStmts.meetsRule[stmtNo] =false;
        }
    }
    return meetsCk;
}

String findURI(RDFStmts rdfStmts, int docNum, InputParameters ip){
//identifies URI (subject of triple where predicate matches criteria)
    String uri = "";
    String designateObj1 = ip.designateObj1;
    String designateObj2 = ip.designateObj2;
```

```
        String designatePred1 = ip.designatePred1;

        String designatePred2 = ip.designatePred2;

        String designatePred3 = ip.designatePred3;

        int stmt = 0;

        boolean found = false;

        while (stmt < rdfStmts.numStmts && !found){

            if ((rdfStmts.docPrd [stmt].equalsIgnoreCase(designatePred1)) ||

                                    (rdfStmts.docPrd [stmt].equalsIgnoreCase(designatePred2)) ||

                                    (rdfStmts.docPrd [stmt].equalsIgnoreCase(designatePred3))){

                found = true;

                uri = rdfStmts.docSub[stmt];

            }

            if ((rdfStmts.docObj[stmt].equalsIgnoreCase(designateObj1) ||

                rdfStmts.docObj[stmt].equalsIgnoreCase(designateObj2))){

                found = true;

                uri = rdfStmts.docSub[stmt];

            }

            stmt++;

        }

        return uri;

    }


    boolean checkPred(String conceptName, InputParameters ip){
```

```
//determines if predicate of triple matches criteria
    boolean retVal = (conceptName.equalsIgnoreCase(ip.pred1a) ||
                    conceptName.equalsIgnoreCase(ip.pred1b) ||
                    conceptName.equalsIgnoreCase(ip.pred1c) ||
                    conceptName.equalsIgnoreCase(ip.pred2a) ||
                    conceptName.equalsIgnoreCase(ip.pred2b)||
                    conceptName.equalsIgnoreCase(ip.pred2c));
    return (retVal);
}


boolean checkConcept(String conceptName, Ontology onto){
//determines if the conceptName String is one of the concepts
//identified from parsed array
    int tc=0;
    boolean found = false;
    while (tc < onto.conceptCt && !found) {
        if ((onto.concepts.uri[tc].equals(conceptName)) ||
                    (onto.concepts.altUri[tc].equals(conceptName)))
        found = true;
        tc++;
    }
    return found;
}
```

```
int findConcept (String conceptName, int sheetNum, Ontology onto) {
//returns the index of the slot in the ontology arrays that designated the document URI
    int tc=0;
    boolean found = false;
        while (tc < onto.conceptCt && !found) {
            if (onto.concepts.uri[tc].equals(conceptName)||
                onto.concepts.altUri[tc].equals(conceptName))
                found = true;
                tc++;
            }
            if (!found)
            return -1;
            else
                return tc-1;
            }

void calcCommon(TermDocumentMatrix tdm, Stats stats, int docUsed){
//calculates the number of common roots between documents
    boolean [ ]annotFound = new boolean[tdm.numRoots];
    boolean [ ]rootFound = new boolean[tdm.numRoots];
    for (int d = 0; d < stats.numUnique; d++) {
        stats.crAra[d] =0;
        stats.curAra[d] =0;
    }
    int d = 0;
    for (int d1 = 0; d1 < docUsed; d1++) {
        if (validSheet[d1]){
            for (int d2 = d1+1; d2 < docUsed; d2++){
                if (validSheet[d2]){
                    for (int rootCt=0;rootCt<tdm.numRoots;rootCt++){
                        rootFound[rootCt]=false;
                        annotFound[rootCt]=false;
                    }
                    int dcnt = 0;
                    int dcnt2 = 0;
                    for (int nr=0; nr < tdm.numTerms; nr++) {
                        int rootndx=tdm.rootIndex[nr];
                        if (tdm.rootInDocument[d1][rootndx]>0 &&
                                        tdm.rootInDocument[d2][rootndx]>0 &&
                                        validSheet[d1] && validSheet[d2] &&
!rootFound[rootndx]){
                            rootFound[rootndx]=true;
                            dcnt++;
                        }
                        if (tdm.rootInDocument[d1][rootndx]==3 &&
                                           tdm.rootInDocument[d2][rootndx]==3 &&
                                           validSheet[d1]  && validSheet[d2] &&
!annotFound[rootndx]) {
                                   annotFound[rootndx]=true;
                                   dcnt2++;
                               }
                           }
                           stats.crAra[d]=dcnt;
                           stats.curAra[d]=dcnt2;
                           d++;
                       }
                   }
               }
           }
}

void calcRelativity(Ontology onto, TermDocumentMatrix tdm, InputParameters ip){
//calculates the how many of the annotations in the document are contained
//or inferred in the roots of the text
    int numDocUsed = ip.docUsed;
    int cnt = 0;
    int numAnnot=0;
    int numAnnotInfer=0;
```

```
        String termRoot="";
        int rootNdx;
        for (int dc=0;dc<numDocUsed;dc++){
            if (validSheet[dc]){
                numAnnot = 0;
                numAnnotInfer = 0;
                for (int tc = 0; tc < onto.conceptCt; tc++){
                    if (docAnnotMatrix[dc][tc] == 1)
                        numAnnot++;
                    if (docAnnotMatrix[dc][tc]==2){
                        numAnnotInfer++;
                    }
                }
                countAnnotsInDoc[dc]=numAnnot;
                countInfersInDoc[dc]=numAnnotInfer;
            }
        }
    }

    int findStatus(int nt, int nd, Ontology onto, String termRoot){
    //determines the status flag for each term in the document,
    // 0 indicating the term is not present in the document,
    // 1 indicating the term is expected from an annotated concept,
    // 2 indicating the term is inferred from the ontology,
    // 3 indicating the term is unexpected
        int status = 3;
        boolean found = false;
        int tc=0;
        while (tc < onto.conceptCt && ! found) {
            if (docAnnotMatrix[nd][tc] >0 && !found) {
                int numRootsInConcept= onto.concepts.conceptData[tc].numWords;
                int cw=0;
                while (cw<numRootsInConcept && !found) {
                    if (docAnnotMatrix[nd][tc] == 1 &&
                                    onto.concepts.conceptData[tc].annotatedRoots[cw].
                                    equalsIgnoreCase(termRoot)){
                        found=true;
                        status = 1;
                    }
                    cw++;
                }
            }
            tc++;
        }
        tc=0;
        while (tc < onto.conceptCt && ! found) {
            if (docAnnotMatrix[nd][tc] >0 && !found) {
                int cw=0;
                while (cw<onto.concepts.conceptData[tc].numInferred && !found) {
                    if
    (onto.concepts.conceptData[tc].inferredRoots[cw].equalsIgnoreCase(termRoot)){
                        found = true;
                        status = 2;
                    }
                    cw++;
                }
            }
            tc++;
        }
        return status;
    }

    void calcParentMatrix(int docNum, int numConcepts, Ancestors ancestors){
    //creates a matrix that shows which the parents of annotated concepts
    //and the count of those concepts
        double sum = 0.0;
        int parentNum = 0;
        int parentNdx = -1;
        int parentInfNdx=-1;
        for (int pc = 0; pc<ancestors.numParents; pc++){
            countParentMatrix[docNum][pc] = 0;
            countParentInfMatrix[docNum][pc] = 0;
        }
        for (int tc = 0; tc<numConcepts;tc++){
            if (docAnnotMatrix[docNum][tc]==1){
                parentNdx = ancestors.conceptToParent[tc];
                countParentMatrix[docNum][parentNdx]++;
            }
            if (docAnnotMatrix[docNum][tc]>=1){
                parentInfNdx = ancestors.conceptToParent[tc];
                countParentInfMatrix[docNum][parentInfNdx]++;
            }
        }
    }
    }


    class ConceptData {
    //This class defines the properties for a specific concept
    int numInferred;
    int numWords;
    String conceptText;
```

```
String[ ] annotatedRoots;
String [ ] inferredRoots;


ConceptData( ){
    System.err.println("Warning--ConceptData instantiated with no size");
    System.err.println("Program will end");
    System.exit(1);
}


ConceptData(int nw, int ni) {
    numWords = nw;
    numInferred = ni;
    annotatedRoots = new String[nw];
    inferredRoots = new String[ni];
    for (int wct=0; wct<nw; wct++){
        annotatedRoots[wct]="";
    }
    for (int ict=0; ict<ni; ict++){
        inferredRoots[ict]="";
    }
}
}
```

```
import java.util.StringTokenizer;
class DocText {
//This class defines the properties of text for a document.
String theText;

DocText(){
    System.err.println("Warning--DocText instantiated with no string");
    System.err.println("Program will end");
    System.exit(1);
}

DocText(String s){
    theText=s.toLowerCase();
}

public Words parseWords (){
//stores the individual words in the text of the concept name in an object of type Words
    StringTokenizer wordstring = new StringTokenizer(theText," \t\n\r.,:/_",false);
    int numWords=wordstring.countTokens();
    int ct=0;
    Words wordList = new Words (numWords);
    while (wordstring.hasMoreTokens()){
        wordList.theWords[ct]=wordstring.nextToken();
        ct++;
    }
    return wordList;
}
}



import java.util.*;
import java.io.*;
class InputParameters{
//This class defines the input parameters file
BufferedReader br;
int maxRDFStmtInOntology;
int maxWordsInDocument;
int maxSuperConceptPerConcept;
int maxLoParent;
String runID;
int maxVal;
int docUsed;
double [ ] selectThresh = new double[4];
String urlUsed;
int numLevelOnes;
int totWords;

int maxStmts;

int max;

int conceptSize;

int minCommon;

int numDeSels;

String rdfConcept;

String rdfConcept2;

String designateTitle;

String designateTitle2;

String pred1a;

String pred1b;

String pred1c;

String pred2a;

String pred2b;

String pred2c;

String daml;

String owl;

String sub;

String designateObj1;

String designateObj2;

String designatePred1;

String designatePred2;
String designatePred3;
String predCond1;
String predCond2;
String owlPredCond;
String cond1;
String altCond1;
String cond2;
String altCond2;


InputParameters(){}
    void readInput(){
//reads the input file containing data concerning ontology
//and the parameters for this particular run
    try{
        br = new BufferedReader(new InputStreamReader
                                                        (new FileInputStream(new
File("inputParams.dat"))));
        numLevelOnes = Integer.parseInt(br.readLine());
        maxVal = Integer.parseInt(br.readLine());
        maxRDFStmtInOntology = Integer.parseInt(br.readLine());
        maxWordsInDocument = Integer.parseInt(br.readLine());
        maxSuperConceptPerConcept = Integer.parseInt(br.readLine());
        maxLoParent = Integer.parseInt(br.readLine());
```

```
maxStmts = Integer.parseInt(br.readLine());
totWords = Integer.parseInt(br.readLine());
conceptSize = Integer.parseInt(br.readLine());
docUsed= Integer.parseInt(br.readLine());
minCommon = Integer.parseInt(br.readLine());
daml= br.readLine();
owl=br.readLine();
urlUsed=br.readLine();
sub=br.readLine();
rdfConcept=br.readLine();
rdfConcept2=br.readLine();
designateTitle=br.readLine();
designateTitle2=br.readLine();
pred1a=br.readLine();
pred1b=br.readLine();
pred1c=br.readLine();
pred2a=br.readLine();
pred2b=br.readLine();
pred2c=br.readLine();
designateObj1=br.readLine();
designateObj2=br.readLine();
designatePred1=br.readLine();
designatePred2=br.readLine();
```

```
designatePred3=br.readLine();
predCond1=br.readLine();
predCond2=br.readLine();
owlPredCond=br.readLine();
cond1=br.readLine();
altCond1=br.readLine();
cond2=br.readLine();
altCond2=br.readLine();
numDeSels=Integer.parseInt(br.readLine());
selectThresh[0] = Double.parseDouble(br.readLine());
selectThresh[1] = Double.parseDouble(br.readLine());
selectThresh[2] = Double.parseDouble(br.readLine());
selectThresh[3] = Double.parseDouble(br.readLine());
String id = br.readLine();
boolean r1,r2;
r1 = (cond1.equals("null")) || (altCond1.equals("null"));
r2 = (cond2.equals("null")) || (altCond2.equals("null"));
String rule="";
if (!r1 && r2)
    rule = "Rule 1 Formal Language";
if (r1 && !r2)
    rule = "Rule 2 Topic";
if (!r1 && !r2)
```

```
                rule = "Rule 3 Both";
          if (!((rule.equals("Rule 1 Formal Language")) || (rule.equals("Rule 2 Topic")) ||
                    (rule.equals("Rule 3 Both")))){
               System.err.println("Error in rule calculation...program will end");
               System.exit(1);
          }
          runID = id + " "+rule+ " NumDeSels"+numDeSels+" MinCommon"+minCommon;
     }
     catch (Exception e){
     System.err.println("Error in reading Input Parameter File");
     System.err.println("Program will end");
     System.err.println(e.getMessage( ));
     e.printStackTrace( );
     System.exit(1);
     }
}
}



import java.sql.*;
import java.io.*;
import sun.jdbc.odbc.*;
import java.text.*;
import java.util.StringTokenizer;

class Ontology{

//This class defines the properties of the ontology in various formats

Connection con;

double [ ] [ ] dotProduct;

int conceptCt;

int numLevelOnes;

int owlRow;

int rowCt;

int size;

String[ ] altObject;

String[ ] altPredicate;

String[ ] altSubject;

String[ ] object;

String[ ] predicate;

String[ ] subject;

String url;

String altCondition1;

String altCondition2;

String altUrl;

String condition1;

String condition2;

String password;
String subConcept;
String user;
String predCondition1;
String predCondition2;
String owlPredCondition;
RDFConceptArray concepts;

Ontology( ){
    System.err.println("***Warning Default Ontology Constructed with no input
                                                        parameters");
    System.err.println("***Program will exit");
    System.exit(1);
}

Ontology(RDFConceptArray conceptsRDF, InputParameters ip){
    size= ip.maxRDFStmtInOntology;
    numLevelOnes = ip.numLevelOnes;
    subject = new String[size];
    predicate = new String[size];
    object = new String[size];
    altSubject = new String[size];
    altPredicate = new String[size];
    altObject = new String[size];
    url=ip.daml;
    altUrl = ip.owl;
    subConcept = ip.sub;
    predCondition1 = ip.predCond1;
    predCondition2 = ip.predCond2;
    owlPredCondition = ip.owlPredCond;
    user = "";
    password = "";
    condition1 = ip.cond1;
    condition2 = ip.cond2;
    altCondition1=ip.altCond1;
    altCondition2=ip.altCond2;
    concepts = conceptsRDF;
}

void readRDF( ){
//utilizes a JdbcOdbc connection to read the spreadsheet files containing
//the data for the DAML version of the ontology
    try{
        new JdbcOdbcDriver();
        Connection con = DriverManager.getConnection(url,user,password);
        Statement st = con.createStatement();
        String query = "SELECT * FROM  [Sheet1$]";
        ResultSet rs = st.executeQuery(query);
        String subj;
        String pred;
        String obj;
        char quote = '\"';
        rowCt = 0;
        while (rs.next()){
            subj = rs.getString(1);
            pred = rs.getString(2);
            obj = rs.getString(3);
            subject[rowCt] = subj;
            predicate[rowCt] = pred;
            object[rowCt] =  obj;
            rowCt++;
        }
    }
    catch(Exception ex) {
        System.err.print("Exception: ");
        System.err.println(ex.getMessage());
        ex.printStackTrace();
        System.err.println("Error reading Ontology file");
        System.exit(1);
    }
```

```
    readRDFowl();
}

void readRDFowl( ){
//utilizes a JdbcOdbc connection to read the spreadsheet files containing
//the data for the OWL version of the ontology
    try{
        new JdbcOdbcDriver();
        Connection con = DriverManager.getConnection(altUrl,user,password);
        Statement st = con.createStatement();
        String query = "SELECT * FROM  [Sheet1$]";
        ResultSet rs = st.executeQuery(query);
        owlRow=0;
        String subject;
        String predicate;
        String object;
        char quote = '\"';
        while (rs.next()){
            subject = rs.getString(1);
            predicate = rs.getString(2);
            object = rs.getString(3);
            altSubject[owlRow] = subject;
            altPredicate[owlRow] = predicate;
            altObject[owlRow] =  object;
            owlRow++;
        }
    }
    catch(Exception ex) {
        System.err.print("Exception: ");
        System.err.println(ex.getMessage());
        ex.printStackTrace();
        System.exit(1);
    }
}

public void findConcepts( ){
// identifies the DAML concepts based on the input conditions
//and calls method to identify OWL concepts.
    conceptCt = 0;
    String predCondition = "type";
    for (int rc=0; rc<rowCt; rc++) {
        if ((subject[rc] != null)&&(predicate[rc] != null) &&          (object[rc] != null) &&
                        ((object[rc].equalsIgnoreCase (condition1)) ||
                        (object[rc].equalsIgnoreCase (condition2))) &&
                        (predicate[rc].endsWith (predCondition))){
            concepts.uri[conceptCt] = subject[rc];
            concepts.name[conceptCt]= findName(conceptCt);
            if (object[rc].equalsIgnoreCase(condition1))
                concepts.branch[conceptCt] = 1;
            else
                concepts.branch[conceptCt] =2;
            conceptCt++;
        }
    }
    findOwlConcepts();
}

void findOwlConcepts( ){
// identifies the OWL concepts based on the input conditions
    String predCondition = "type";
    for (int rc=0; rc<owlRow; rc++) {
        if ((altSubject[rc] != null)&&(altPredicate[rc] != null) && (altObject[rc] != null)
&&
                            ((altObject[rc].equalsIgnoreCase (altCondition1)) ||
                            (altObject[rc].equalsIgnoreCase (altCondition2))) &&
                            (altPredicate[rc].endsWith (predCondition))){
            String owlName = findName(altSubject[rc]);
            if (owlName.length() < 1){
                System.err.println("Name not found for altObject[rc]");
                System.exit(1);
            }
            boolean found=false;
            int damlCt=0;
            while (damlCt<conceptCt && !found){
                if (concepts.name[damlCt].equalsIgnoreCase(owlName)){
                    found=true;
                    concepts.altUri[damlCt] = altSubject[rc];
                }
                damlCt++;
            }
            if (!found) {
                System.err.println("ERROR IN NAME RESOLUTION");
                System.exit(1);
            }
        }
    }
}

void findConceptsLists( ) {
//transforms data in concept lists to Words objects representing the words actually
//annotated and additional words which can be inferred from ontology
    String theText, theWord;
    String inferredText="";
    Roots rootWord = new Roots();
    Words iRootsFullString;
    String [ ] infRootWords = new String [conceptCt*conceptCt];
    for (int tc=0; tc<conceptCt; tc++){
        theText= concepts.name[tc];
        StringTokenizer wordString = new StringTokenizer(theText," \t\n\r.,:/_#",false);
        int numWords=wordString.countTokens();
        Words rootsFullString = new Words(numWords);
        int ct=0;
        while (wordString.hasMoreTokens()){
            rootsFullString.theWords[ct]=wordString.nextToken();
            ct++;
        }
        Stop wordStop = new Stop( );
        wordStop.elimStops(rootsFullString);
        for (int wc=0;wc<rootsFullString.wordCt;wc++){

    rootsFullString.theWords[wc]=rootWord.findRoot(rootsFullString.theWords[wc]);
        }
        rootsFullString.sortWords(0,rootsFullString.wordCt-1);
        rootsFullString.elimDups( );
        String checkRoot="";
```

```
        int infCt = 0;
        if (concepts.numSuperConcepts[tc]>0){
            for (int stc=0; stc<concepts.numSuperConcepts[tc]; stc++){
                StringTokenizer infString =
                                  new StringTokenizer(concepts.superList[tc][stc],"
\t\n\r.,:/_#",false);
                while (infString.hasMoreTokens()){
                    checkRoot=rootWord.findRoot(infString.nextToken());
                    if (!rootExists(checkRoot,rootsFullString.theWords,
rootsFullString.wordCt)) {
                        if (!rootExists(checkRoot,infRootWords,infCt)) {
                            infRootWords[infCt]=checkRoot;
                            infCt++;
                        }
                    }
                }
            }
            Words rootsInferred = new Words(infCt);
            rootsInferred.wordCt = infCt;
            rootsInferred.theWords=infRootWords;
            rootsInferred.sortWords(0,infCt-1);
            concepts.conceptData[tc] =
                              new ConceptData(rootsFullString.wordCt,rootsInferred.wordCt);
            concepts.conceptData[tc].numWords=rootsFullString.wordCt;
            concepts.conceptData[tc].annotatedRoots=rootsFullString.theWords;
            concepts.conceptData[tc].numInferred = rootsInferred.wordCt;
            concepts.conceptData[tc].inferredRoots=rootsInferred.theWords;
        }
        else {
            concepts.conceptData[tc] = new ConceptData(rootsFullString.wordCt,0);
            concepts.conceptData[tc].numWords=rootsFullString.wordCt;
            concepts.conceptData[tc].annotatedRoots=rootsFullString.theWords;
            concepts.conceptData[tc].numInferred=0;
        }
    }
}

boolean rootExists(String root,String [ ] rootWords, int ct){
//determines if a root already exists in the Words object
    boolean found = false;
    int wc = 0;
    while (wc < ct) {
        if (rootWords[wc].equalsIgnoreCase(root))
            found = true;
        wc++;
    }
    return found;
}


String findName(int tc ){
//determines the literal name associated with an ontology concept,
//based on the position of the concept the DAML ontology arrays
//name will default to the concept at the tc location in ontology URI array
//if no literal name is found
    String retVal="";
    boolean found=false;
    int rc = 0;
    while (rc<rowCt && ! found) {
        boolean b1=false;
        boolean b2a=false;
        boolean b2b=false;
        if (subject[rc]!=null)
            b1=subject[rc].equalsIgnoreCase (concepts.uri[tc]);
        int lpc=predCondition1.length() ;
        int predLen=predicate[rc].length();
        int start=predLen-lpc;
        String ending;
        if (predLen>=lpc){
            ending=predicate[rc].substring(start,predLen);
        }
        else
            ending="";
        ending=ending.toLowerCase();
        predCondition1=predCondition1.toLowerCase();
        predCondition2=predCondition2.toLowerCase();
        if (ending!=null){
            b2a=ending.equalsIgnoreCase(predCondition1);
            b2b=ending.equalsIgnoreCase(predCondition2);
        }
        boolean test1= ((subject[rc] != null)&&(predicate[rc] != null) &&
                        (object[rc] != null) &&  (b1)   && (b2a) );
        boolean test2= ((subject[rc] != null)&&(predicate[rc] != null) &&
                        (object[rc] != null) &&  (b1)   && (b2b) );
        boolean test = test1 || test2; //allow names from either DAML or OWL ontologies
        if (test){
            retVal=object[rc].substring(1,object[rc].length()-1);
            if (b2b)
                retVal = findLiteral(object[rc]);
            found = true;
        }
        rc++;
    }
    if (!(found)) retVal = concepts.uri[tc];
        return retVal;
}


String findName(String conceptUri ){
//determines the literal name associated with an ontology concept,
//based on the String meeting the concept conditions in the OWL ontology
//name will default to the input String if no literal name is found
    String objCondition = "";
    String retVal="";
    boolean found=false;
    int rc = 0;
    while (rc<owlRow && ! found) {
        boolean b1 = false;
        boolean b2 = false;
        boolean test = false;
        if (altSubject[rc]!=null)
            b1=altSubject[rc].equalsIgnoreCase(conceptUri);
        if (altPredicate[rc]!=null)
            b2 = altPredicate[rc].endsWith (owlPredCondition);
        test = ((altSubject[rc] != null)&&(altPredicate[rc] != null) &&
                        (altObject[rc] != null) &&  (b1)   && (b2) );
        if (test){
            retVal=findLiteral(altObject[rc]);
            found = true;
        }
        rc++;
    }
        if (!(found)) retVal = conceptUri;
            return retVal;
}

String findLiteral(String litObject){
//strips parentheses from the literal String to return the name specified
    char openParen='(';
    char closeParen=')';
    int startPos = litObject.indexOf(openParen)+1;
    String answer = litObject.substring(startPos,litObject.length()-1);
    return answer;
}

void findLevels ( ){
//finds the level number of concepts
//input conditions specified to target concepts of interest are designated
//level 1 and other level numbers are calculated based on distance from those concepts in
//the ontology tree by calculated the number of superconcepts between them
```

```
    int nc=1;
    int lvl;
    for (int tc = 0; tc<conceptCt; tc++){
        lvl = findLevelThisConcept (tc,nc);
        concepts.lowestLevel[tc] = lvl;
    }
}

int findLevelThisConcept(int conceptNdx, int level){
//finds the level number of a specific concepts
    int newNdx;
    int thisNdx= conceptNdx;
    while (hasSuperConcept(thisNdx)){
        level++;
        newNdx=findSuperConcept(thisNdx);
        thisNdx=newNdx;
    }
return level;
}

boolean hasSuperConcept(int ndx) {
//determines if the concept in the ontolgoy arrays at position
//ndx has a superConcept
```

```
    String name = concepts.uri[ndx];
    boolean answer = false;
    int tc = 0;
    while ((tc<rowCt-1) && !(answer)){
        if (((predicate[tc].endsWith(subConcept))) &&
                        ((concepts.uri[ndx].equalsIgnoreCase(object[tc])))){
            answer = true;
        }
        tc++;
    }
return answer;
}

int findSuperConcept(int ndx) {
//returns the index number of the superConcept for the concept
//in the ontology arrays at position ndx
    String name = concepts.uri[ndx];
    boolean answer = false;
    int tc = 0;
    int ndxSuperConcept=-1;
    while ((tc<rowCt) && !(answer)){
        if ((concepts.uri[ndx].equalsIgnoreCase(object[tc])) &&
                        (predicate[tc].endsWith(subConcept))) {
```

```
            ndxSuperConcept=findNdxSuperConcept(subject[tc]);
            answer = true;
        }
        tc++;
    }
    return ndxSuperConcept;
}

int findNdxSuperConcept(String sub) {
//returns the index number of the superConcept for the concept
//specified by the String sub
    int retVal = -1;
    int n = 0;
    boolean found = false;
    while ((n<conceptCt) && !(found)) {
        if (concepts.uri[n].equalsIgnoreCase(sub)) {
            retVal = n;
            found = true;
        }
        else n++;
    }
    return retVal;
}
```

```
int findNdxSuperConcept(String sub, int val) {
//returns the index number of the superConcept for the concept
//specified by the String sub and integer val
    int retVal = -1;
    int n=0;
    boolean found=false;
    while ((n<conceptCt) && !(found)) {
    if (concepts.name[n].equalsIgnoreCase(sub)) {
        retVal = n;
        found= true;
    }
    else n++;
    }
    return retVal;
}

void calcTermSim(Ancestors ancestors) {
//calculates the similarity of annotations for the concepts
    dotProduct = new double [conceptCt] [conceptCt];
    for (int rc = 0; rc < conceptCt; rc++)
        for (int cc = 0; cc < conceptCt; cc++){
            dotProduct[rc] [cc] = 0.0;
    }
    for (int rc = 0; rc < conceptCt; rc++)
        for (int cc = 0; cc <= rc; cc++){
            if (rc == cc)
                dotProduct [rc] [cc] = 1.0;
            else
                dotProduct[rc] [cc] = (2.0 * ancestors.commonAncestorArray[rc][cc]) /
                                    (double)(concepts.lowestLevel[rc] *
concepts.lowestLevel[cc]);
        }
}

void findSuperConcepts( ) {
//identifies the superConcepts of each concept and assigns the appropriate cells
//of the superMatrix to 1 to denote that a superConcept exists for that concept
    try{
    for (int tc = 0; tc< conceptCt; tc++){
        for (int rc = 0; rc< rowCt; rc++){
            if (object[rc] != null){
                if ((concepts.uri[tc].equalsIgnoreCase(object[rc]))
                                    && (predicate[rc].endsWith(subConcept))) {
                    concepts.superList[tc][concepts.numSuperConcepts[tc]] =
                                        findSuperConceptName(subject[rc]);
                    int ndx = findNdxSuperConcept(subject[rc]);
                    concepts.superMatrix[tc][ndx]=1;
                    concepts.numSuperConcepts[tc]++;
                }
            }
        }
    }
    completeSuperConceptList();
    createLowestLevelParentList( );
    }
    catch(Exception e){
    System.err.println("exception " + e);
    e.printStackTrace();
    System.exit(1);
    }
}

void completeSuperConceptList(){
//adds the superConcept names to the SuperLists and
//calls method to add all Parents of that concept
    String name;
    int ndx;
    int index = -1;
    for (int   tc = 0; tc< conceptCt; tc++){
        int lastDone = 0;
        int previousEnd =concepts.numSuperConcepts[tc];
        while (lastDone < previousEnd) {
            previousEnd =concepts.numSuperConcepts[tc];
            for (int st = lastDone; st<concepts.numSuperConcepts[tc]; st++){
            name =  concepts.superList[tc][st];
            index = findNdxSuperConcept(name,0);
            if (hasSuperConcept(index))
                addAllParents(index,tc);
            }
            lastDone = previousEnd;
        }
    }
}

void createLowestLevelParentList( ){
//declares a list of the lowest level of each parent  of each concept
    String name="";
    int index;
    for (int   tc = 0; tc< conceptCt; tc++){
        int parentNum = 0;
        boolean found;
```

```
if (concepts.numSuperConcepts[tc] > 0) {
    for (int st = 0; st<concepts.numSuperConcepts[tc]; st++) {
        name =  concepts.superList[tc][st];
        index = findNdxSuperConcept(name,0);
        found = false;
        while (hasSuperConcept(index))
            index=findSuperConcept(index);
        if (parentNum <1) {
            concepts.lowestParent[tc][parentNum]=index;
            parentNum++;
        }
        else{
            for (int prevParent=0;prevParent<parentNum;prevParent++){
                if (index==concepts.lowestParent[tc][prevParent])
                found = true;
            }
            if (!found){
                concepts.lowestParent[tc][parentNum]=index;
                parentNum++;
            }
        }
    }
}
```

```
        else {
            concepts.lowestParent[tc][parentNum]=tc;
        }
    }
}

void addAllParents(int ndx, int tc){
//declares a list of all parents (superConcepts) of each concept
    for (int rc = 0; rc< rowCt; rc++){
        boolean answer = false;
        if (((predicate[rc].endsWith(subConcept))) &&
                    ((concepts.uri[ndx].equalsIgnoreCase(object[rc])))){
            answer = true;
            int index = findNdxSuperConcept(subject[rc]);
            String name = concepts.name[index];
            if (!checkDupSuperList(tc,name)){
                concepts.numSuperConcepts[tc]++;
                concepts.superMatrix[tc] [index] = 1;
                concepts.superList[tc][concepts.numSuperConcepts[tc]-1]=name;
            }
        }
    }
}
```

```
boolean checkDupSuperList(int tc, String name) {
//checks for duplicate entries in the superList
    boolean found = false;
    int max = concepts.numSuperConcepts[tc];
    for (int rc=0; rc > max; rc++)
        if (name.equals(concepts.superList[tc][rc]))
            found = true;
    return found;
}

String findSuperConceptName(String uri){
//returns the String name of the superConcept  given the URI
    String retVal = "";
    boolean found=false;
    int tc=0;
    while (tc<conceptCt && !found){
        if (concepts.uri[tc].equalsIgnoreCase(uri)){
            found=true;
            retVal = findName(tc);
        }
        tc++;
    }
    if (!found)   {
        System.err.println("Warning--Did not find uri of SuperConcept--End Pgm");
        System.exit(1);
    }
    return retVal;
}
}


class RDFConceptArray {
//This class defines the concepts found in the ontology and their superconcepts
ConceptData [ ] conceptData;
int [ ] branch;
int [ ] lowestLevel;
int [ ] [ ] lowestParent;
int [ ] numSuperConcepts;
int [ ] [ ] superMatrix;
int [ ] [ ] superNdx;
int maxLowestParents;
int maxSuperConcepts;
int size;
String [ ] altUri;
String[ ] uri;
```

```
String [ ] [ ] firstSuperUri;
String[ ] name;
String [ ] [ ] superList;

RDFConceptArray ( ) {
    System.err.println("RDFConceptArray default construction--no parameters
specified—
                                                        Program will end");
    System.exit(1);
}

RDFConceptArray(InputParameters ip){
    size = ip.maxRDFStmtInOntology;
    maxSuperConcepts = ip.maxSuperConceptPerConcept;
    maxLowestParents = ip.maxLoParent;
    uri = new String [size];
    altUri = new String [size];
    name = new String[size];
    branch = new int [size];
    lowestLevel = new int [size];
    numSuperConcepts = new int [size];
    superNdx = new int [size] [size];
    superMatrix = new int [size] [size];
    lowestParent = new int [size] [maxLowestParents];
    conceptData = new ConceptData[size];
    firstSuperUri = new String[size] [size];
    superList = new String [size] [maxSuperConcepts];
    for (int concept = 0; concept < size; concept++){
        numSuperConcepts[concept] = 0;
        branch[concept] = 0;
        lowestLevel[concept] = 0;
        for (int rc=0; rc<size; rc++) {
            superMatrix[concept] [rc] = 0;
            superNdx[concept] [rc] = 0;
        }
        for (int sconcept = 0; sconcept < maxSuperConcepts; sconcept ++) {
            superList[concept][sconcept] = "";
        }
        for (int hPar=0;hPar<maxLowestParents;hPar++){
            lowestParent[concept][hPar]=-1;
        }
    }
}

int lookUpLevel(String s, int num){
//determines the level for a concept, given the URI
```

```
    int retVal = -1;
    for (int ln=0;ln<num;ln++){
        if (name[ln].equals(s))
            retVal=lowestLevel[ln];
    }
    return retVal;
}

int lookUpIndex(String s, int num){
//determines the index for a concept, given the URI
    int retVal = -1;
    for (int ln=0;ln<num;ln++){
        if (name[ln].equals(s))
            retVal=ln;
        }
        return retVal;
    }
}



class RDFStmts {
//This class defines the properties of specific RDF statements
boolean [ ] meetsRule;
```

```
int numStmts;
String [ ] docObj;
String [ ] docPrd;
String [ ] docSub;

RDFStmts( ){
    System.err.println("***Warning Default RDFStmt Constructed without # of
                                                statements");
    System.err.println("***Program will exit");
    System.exit(1);
}

RDFStmts (int max) {
    docSub = new String [max];
    docObj = new String [max];
    docPrd = new String [max];
    meetsRule = new boolean [max];
    numStmts = 0;
}
}

class Roots{
```

```
//This class defines properities of word roots and their calculation


Roots() {}


String findRoot(String term){

//finds the root of a term by eliminating suffixes and final

//e, y or double letter

    String[ ] suffixList7 = {"ability","itional"};

    String[ ] suffixList6 = {"atings","ations","ically", "itions"};

    String[ ] suffixList5 =

                        {"ables","ating","ation","ators","ences","ities",

                           "ition","ments","tions"};

    String[ ] suffixList4 = {"able","ally","ator",

                           "ings","ives","ment","tion"};

    String[ ] suffixList3 = {"ent","ers","ics","ied",

                           "ies","ily","ing","ity","ive","ors"};

    String[ ] suffixList2 = {"ed", "er","es","ic","ly","or"};

    String[ ] suffixList1 = {"s","y"};

    char first=term.charAt(0);

    int val = Character.getNumericValue(first);

    String root = term;

    int ndx;

    String apos = "'";
    String hyphen = "-";
    char firstChar=root.charAt(0);
    while (Character.getNumericValue(firstChar)<0 ){
        root=root.substring(1,root.length());
        firstChar = root.charAt(0);
    }
    ndx = root.lastIndexOf(apos);
    if (ndx>-1)
        root=root.substring(0,ndx);
    ndx= root.indexOf(hyphen);
    int cnt = 0;
    boolean found = false;
    while (cnt<suffixList7.length && ! found && root.length( )>8) {
        if (root.endsWith(suffixList7[cnt])) {
            root = root.substring(0,root.length( )-7);
            found = true;
        }
        cnt++;
    }
    cnt = 0;
    found = false;
    while (cnt<suffixList6.length && ! found && root.length( )>7) {
        if (root.endsWith(suffixList6[cnt])) {
            root = root.substring(0,root.length( )-6);
            found = true;
        }
        cnt++;
    }
    cnt = 0;
    found = false;
    while (cnt<suffixList5.length && ! found && root.length( )>6) {
        if (root.endsWith(suffixList5[cnt])) {
            root = root.substring(0,root.length( )-5);
            found = true;
        }
        cnt++;
    }
    cnt = 0;
    found=false;
    while (cnt<suffixList4.length && ! found && root.length( )>5) {
        if (root.endsWith(suffixList4[cnt])) {
            root = root.substring(0,root.length( )-4);
            found = true;
        }
        cnt++;
    }
```

```
cnt = 0;
found = false;
while (cnt<suffixList3.length && ! found && root.length( )>4) {
    if (root.endsWith(suffixList3[cnt])) {
    root = root.substring(0,root.length( )-3);
        found = true;
    }
    cnt++;
}
cnt = 0;
found=false;
while (cnt<suffixList2.length && ! found && root.length( )>3) {
    if (root.endsWith(suffixList2[cnt])) {
        root = root.substring(0,root.length( )-2);
        found = true;
    }
    cnt++;
}
cnt = 0;
found = false;
while (cnt<suffixList1.length && ! found && root.length( )>2) {
    if (root.endsWith(suffixList1[cnt])) {
        root = root.substring(0,root.length( )-1);
```

```
            found = true;
        }
        cnt++;
    }
    char last;
    char ntl = ' ';
    last = root.charAt(root.length( )-1);
        if (root.length( )>2)
            ntl = root.charAt(root.length( )-2);
        while ((root.length() > 2) && ((last == 'e') || (last == 'y'))){
            root=root.substring(0,root.length()-1);
            last = root.charAt(root.length( )-1);
            if (root.length( )>2)
                ntl = root.charAt(root.length( )-2);
    }
    while ((root.length() > 1) && (last == ntl)){
    root=root.substring(0,root.length()-1);
    last = root.charAt(root.length( )-1);
    if (root.length( )>1)
        ntl = root.charAt(root.length( )-2);
    }
    return root;
}
```

```
}

import java.io.*;
class Stats{
//This class models the statistics needed to analyze
//similarity between annotations and text of documents
double [ ] whiCk = new double[4];
double [ ] whiInfCk = new double[4];
boolean [ ] selectPair;
double tol = 1.e-14;
double [ ] simAra;
double [ ] simInfAra;
double [ ] jacAra;
double [ ] jacInfAra;
double [ ] vecAra;
double [ ] vecInfAra;
double [ ] gcsmAra;
double [ ] uhiAra;
double [ ] whiAra;
double [ ] simRankAra;
double [ ] simInfRankAra;
double [ ] jacRankAra;
double [ ] jacInfRankAra;
double [ ] vecRankAra;
double [ ] vecInfRankAra;
double [ ] gcsmRankAra;
double [ ] uhiRankAra;
double [ ] whiRankAra;
double ratiobe [ ];
double ratiopbe [ ];
double normDiff[ ];
double avgDiff[ ];
double simCc;
double simInfCc;
double jacCc;
double jacInfCc;
double vecCc;
double vecInfCc;
double gcsmCc;
double uhiCc;
double whiCc;
double simRCc;
double simInfRCc;
double jacRCc;
double jacInfRCc;
double vecRCc;
double vecInfRCc;
double gcsmRCc;
double uhiRCc;
double whiRCc;
int [ ] crAra;
int [ ] curAra;
int [ ] ndx1CrAra;
int [ ] ndx2CrAra;
int [ ] ndx1CurAra;
int [ ] ndx2CurAra;
double [ ]  rankCr;
double [ ]  rankCUR;
int [ ] either;
int [ ] eitherInf;
int [ ] both;
int [ ] bothInf;
int [ ] numSameParentEither;
int [ ] numSameParentBoth;
int statMax;
int docCheck=2;
int maxDeSels=10;
int e;
int aNotb;
int bNota;
int neither;
int b;
int eInf;
int aNotbInf;
int bNotaInf;
int neitherInf;
int bInf;
int pe;
int numUnique;
int pb;
double dot;
double pDot;
double wpDot;
double sA;
double sB;
double psA;
double psB;
double wpsA;
double wpsB;
int ndxPair[ ] [ ];
```

```
Stats(){
    System.err.println("Stats default construction--no numUnique specified");
    System.exit(1);
}


Stats(Annotations annot, InputParameters ip) {
    statMax = ip.maxVal;
    numUnique = ip.docUsed * (ip.docUsed-1) / 2;
    int numDocs = ip.docUsed;
    ndxPair = new int[numDocs] [numDocs];
    selectPair = new boolean[numUnique];
    crAra = new int[numUnique];
    curAra = new int[numUnique];
    rankCr = new double[numUnique] ;
    rankCUR = new double[numUnique];
    simAra = new double [numUnique] ;
    simInfAra = new double [numUnique] ;
    simRankAra = new double [numUnique];
    simInfRankAra = new double [numUnique];
    jacAra = new double [numUnique] ;
    jacInfAra = new double [numUnique] ;
    jacRankAra = new double [numUnique] ;
    jacInfRankAra = new double [numUnique] ;
    vecAra = new double [numUnique];
    vecInfAra = new double [numUnique];
    vecRankAra = new double [numUnique];
    vecInfRankAra = new double [numUnique];
    gcsmAra = new double [numUnique];
    gcsmRankAra = new double [numUnique];
    uhiAra = new double [numUnique];
    uhiRankAra = new double [numUnique];
    whiAra = new double [numUnique];
    whiRankAra = new double [numUnique];
    either = new int [numUnique];
    eitherInf = new int [numUnique];
    both = new int [numUnique] ;
    bothInf = new int [numUnique] ;
    ratiobe = new double [numUnique];
    ratiopbe = new double [numUnique];
    numSameParentEither = new int [numUnique];
    numSameParentBoth = new int [numUnique] ;
    normDiff = new double[numUnique];
    avgDiff = new double[numDocs];
}

void calcSimilarity(Annotations annot, Ancestors ancestors,
                                          Ontology onto, String runID, int numDocs,
double [ ] th,
                                          TermDocumentMatrix tdm, int numDeSels, int
minCommon){
//calls other methods to calculate the various measures of similarity,
//calculate correlation and write output file for each case
    indexDocs(numDocs, annot);
    int caseNo;
    caseNo = 0;
    System.out.println("\n\nCase 0-based on common roots document ratio");
    calcRatios(annot, ancestors, onto, numDocs);
    selectBestDocs(ratiobe,crAra, annot.selectedSheet,  numDocs, th[caseNo],
    annot.validSheet, caseNo, numDeSels, minCommon);
    calcMeasures(annot, ancestors, onto, numDocs, caseNo);
    initRanks(caseNo);
    calcRanks(caseNo,numDocs,annot);
    calcCorr(caseNo,numDocs,annot.selectedSheet);
    writeFile(runID, numDocs, annot, tdm, "DocCR", caseNo, th);
    if (ancestors.hier) {
      System.out.println("\nCase 1-based on common roots parent ratio");
      caseNo=1;
      calcRatios(annot, ancestors, onto, numDocs);
      selectBestDocs(ratiopbe,crAra, annot.selectedSheet,  numDocs, th[caseNo],
                                          annot.validSheet, caseNo, numDeSels,
minCommon);
        calcMeasures(annot, ancestors, onto, numDocs, caseNo);
        initRanks(caseNo);
        calcRanks(caseNo,numDocs,annot);
        calcCorr(caseNo,numDocs,annot.selectedSheet);
        writeFile(runID, numDocs, annot, tdm, "ParentCR",caseNo,th);
    }
    System.out.println("\nCase 2-based on common unexpected roots document ratio");
    caseNo = 2;
    calcRatios(annot, ancestors, onto, numDocs);
    selectBestDocs(ratiobe,curAra, annot.selectedSheet,  numDocs, th[caseNo],
                                         annot.validSheet, caseNo, numDeSels,
minCommon);
    calcMeasures(annot, ancestors, onto, numDocs, caseNo);
i   nitRanks(caseNo);
    calcRanks(caseNo,numDocs,annot);
    calcCorr(caseNo,numDocs,annot.selectedSheet);
    writeFile(runID, numDocs, annot, tdm, "DocCUR",caseNo,th);
    if (ancestors.hier){
        System.out.println("\nCase 3-based on common unexpected roots parent ratio");
        caseNo = 3;
        calcRatios(annot, ancestors, onto, numDocs);
        selectBestDocs(ratiopbe,curAra, annot.selectedSheet,  numDocs, th[caseNo],
                                            annot.validSheet, caseNo, numDeSels,
minCommon);
        int selCt=0;
        calcMeasures(annot, ancestors, onto, numDocs, caseNo);
        initRanks(caseNo);
        selCt=0;
        calcRanks(caseNo,numDocs,annot);
        calcCorr(caseNo,numDocs,annot.selectedSheet);
        writeFile(runID, numDocs, annot, tdm, "ParentCUR",caseNo,th);
    }
}

void indexDocs(int numDocs, Annotations annot){
//calculates the indices for the document pairs
    int docPair=0;
    for (int docA = 0; docA < numDocs; docA++){
        ndxPair[docA][docA]=-1;
        for (int docB = docA+1; docB<numDocs; docB++){
            if (annot.validSheet[docA] && annot.validSheet[docB]) {
                ndxPair[docA][docB] = docPair;
                ndxPair[docB][docA]=docPair;
                docPair++;
            }
            else{
                ndxPair[docA][docB] = -1;
                ndxPair[docB][docA] = -1;
            }
        }
    }
}

void calcRatios(Annotations annot, Ancestors ancestors, Ontology onto,  int numDocs){
//calculates ratios of common annotations in both document pairs
//to annotions in either document
    int docPair=0;
    for (int docA = 0; docA < numDocs; docA++){
        for (int docB = docA+1; docB<numDocs; docB++){
            docPair = ndxPair[docA][docB];
            if (docPair>=0){
                if (annot.validSheet[docA] && annot.validSheet[docB]) {
                    e = findCountInEither(docA,docB,annot,onto.conceptCt, false);
                    b = findCountInBoth(docA, docB,annot,onto.conceptCt, false);
                    pe = findParentCountInEither(docA,docB,annot,onto);
                    pb = findParentCountInBoth(docA,docB,annot,onto);
                    either[docPair] = e;
                    both[docPair] = b;
                    numSameParentEither[docPair]= pe;
                    numSameParentBoth[docPair] = pb;
                    if (e > 0)
                        ratiobe [docPair] = (double)b/(double)e;
                    else
                        ratiobe[docPair] = -1;
                    if (pe > 0) {
                        ratiopbe [docPair] = (double)pb/(double)pe;
                    }
                    else {
                        ratiopbe [docPair] = -1;
                    }
                }
            }
        }
    }
}

void calcMeasures(Annotations annot, Ancestors ancestors, Ontology onto,
                                        int numDocs, int caseNo){
//calculates the various measures of similarity
    int docPair=0;
    for (int docA = 0; docA < numDocs; docA++){
        for (int docB = docA+1; docB<numDocs; docB++){
        docPair = ndxPair[docA][docB];
        if ((docPair>=0)&& (selectPair[docPair])){
            if (annot.selectedSheet[docA] && annot.selectedSheet[docB]) {
                boolean inferred=false;
                aNotb =
findCountIn1stDocNot2nd(docA,docB,annot,onto.conceptCt,inferred);
                bNota=  findCountIn2ndDocNot1st(docA,docB,annot,onto.conceptCt,
inferred);
                neither = findCountInNeither(docA,docB,annot,onto.conceptCt, inferred);
                e = findCountInEither(docA,docB,annot,onto.conceptCt, inferred);
                b = findCountInBoth(docA, docB,annot,onto.conceptCt, inferred);
                pe = findParentCountInEither(docA,docB,annot,onto);
                pb = findParentCountInBoth(docA,docB,annot,onto);
                inferred=true;
                aNotbInf =
findCountIn1stDocNot2nd(docA,docB,annot,onto.conceptCt,inferred);
                bNotaInf =  findCountIn2ndDocNot1st(docA,docB,annot,onto.conceptCt,
inferred);
                neitherInf = findCountInNeither(docA,docB,annot,onto.conceptCt, inferred);
                eInf = findCountInEither(docA,docB,annot,onto.conceptCt, inferred);
                bInf = findCountInBoth(docA, docB,annot,onto.conceptCt, inferred);
```

```
either[docPair] = e;
both[docPair] = b;
eitherInf[docPair] = eInf;
bothInf[docPair] = bInf;
numSameParentEither[docPair]= pe;
numSameParentBoth[docPair] = pb;
if (e > 0)
    ratiobe [docPair] = (double)b/(double)e;
else
    ratiobe[docPair] = -1;
if (pe > 0)
    ratiopbe [docPair] = (double)pb/(double)pe;
else
    ratiopbe [docPair] = -1;
double vA=b;
double vB=bNota;
double vC=aNotb;
double vD=neither;
double t=vA+vB+vC+vD;
if (t>0)
    simAra[docPair] =  (((vA+vB)*(vA+vC))+((vD+vB)*(vD+vC)))/ (t*t);
else
    simAra[docPair]= -1;
```

```
vA=bInf;
vB=bNotaInf;
vC=aNotbInf;
vD=neitherInf;
t=vA+vB+vC+vD;
if (t > 0)
    simInfAra[docPair] = (((vA+vB)*(vA+vC))+((vD+vB)*(vD+vC)))/ (t*t);
else
    simInfAra[docPair] = -1;
if (e >  0){
    jacAra [docPair] = (double) b/ (double)e;
}
else {
    jacAra [docPair] = -1;
}
if (eInf >  0){
    jacInfAra [docPair] = (double) bInf/ (double)eInf;
}
else {
    jacInfAra [docPair] = -1;
}
inferred=false;
dot = findDotProd(docA, docB, annot, onto.conceptCt, inferred);
```

```
sA = findSqrtSumSqr(docA, annot, onto.conceptCt,inferred);

sB = findSqrtSumSqr(docB, annot, onto.conceptCt,inferred);

if ((sA*sB) > 0)

    vecAra[docPair]  = dot / (sA * sB);

else

    vecAra[docPair] = -1;

inferred=true;

dot = findDotProd(docA, docB, annot, onto.conceptCt, inferred);

sA = findSqrtSumSqr(docA, annot, onto.conceptCt,inferred);

sB = findSqrtSumSqr(docB, annot, onto.conceptCt,inferred);

if ((sA*sB) > 0)

    vecInfAra [docPair]  = dot / (sA * sB);

else

vecInfAra [docPair]  = -1;

double g1 = calcGcsm(docA,docB,annot,ancestors,onto);

double gA = calcGcsm(docA,docA,annot,ancestors,onto);

double gB = calcGcsm(docB,docB,annot,ancestors,onto);

double gAB = Math.sqrt(gA) * Math.sqrt(gB);

if (gAB > 0){

    gcsmAra [docPair] = g1 / gAB;

}

else {

    gcsmAra [docPair] = -1;
```

```
        }
        pDot = findPDotProd(docA, docB, annot, ancestors);
        psA = findPSqrtSumSqr(docA, annot, ancestors);
        psB = findPSqrtSumSqr(docB, annot, ancestors);
        if ((psA*psB)>0)
            uhiAra [docPair]  = pDot / (psA * psB);
        else
            uhiAra [docPair]  = -1;
        inferred=true;
        wpDot = findWpDotProd(docA, docB, annot, ancestors, onto.conceptCt);
        wpsA = findWpSqrtSumSqr(docA, annot, ancestors);
        wpsB = findWpSqrtSumSqr(docB, annot, ancestors);
        if ((wpsA * wpsB)>0)
            whiAra [docPair] = wpDot / (wpsA * wpsB);
        else
            whiAra [docPair] = -1;
    }
    else { //one or both of the documents does not have annots of interest
        simAra [docPair] = -1;
        simInfAra [docPair] = -1;
        jacAra [docPair] = -1;
        jacInfAra [docPair] = -1;
        vecAra [docPair] = -1;
```

```
                vecInfAra [docPair] = -1;
                gcsmAra[docPair] = -1;
                uhiAra [docPair] = -1;
                whiAra [docPair] = -1;
                }
            }
        }
    }
    validate(ancestors.hier,numDocs,caseNo);
    normalize (simAra, numDocs);
    normalize (simInfAra, numDocs);
    normalize (jacAra, numDocs);
    normalize (jacInfAra, numDocs);
    normalize (vecAra, numDocs);
    normalize (vecInfAra, numDocs);
    normalize (gcsmAra, numDocs);
    normalize (uhiAra, numDocs);
    normalize (whiAra, numDocs);
}

void normalize (double valAra[ ], int num){
//normalizes valAra based on maximum value in array
    double maxVal = findMaxSel(valAra,num);
    int docPair;
    for (int dA=0;dA<num;dA++){
        for (int dB=dA+1;dB<num;dB++){
            docPair=ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
                    valAra[docPair]=valAra[docPair]/maxVal;
                }
            }
        }
    }
}

double findMinSel(double [ ]valAra, int num){
//returns minimum value in valAra of those documents in selection set
double min=999999;
int docPair;
for (int dA=0;dA<num;dA++){
for (int dB=dA+1;dB<num;dB++){
docPair=ndxPair[dA][dB];
if (docPair >=0){
if (selectPair[docPair]){
if (valAra[docPair]<min){
```

```
min=valAra[docPair];
}
}
}
}
}
return min;
}

double findMaxSel(double [ ]valAra, int num){
//returns maximum value in valAra of those documents in selection set
    double max=-999;
    int docPair;
    for (int dA=0;dA<num;dA++){
        for (int dB=dA+1;dB<num;dB++){
            docPair=ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
                    if (valAra[docPair]>max){
                        max=valAra[docPair];
                    }
                }
            }
        }
    }
return max;
}


double findMinSel(int [ ]valAra, int num){
//returns integer minimum value in valAra of those documents in selection set
    double min=999999;
    int docPair;
    for (int dA=0;dA<num;dA++){
        for (int dB=dA+1;dB<num;dB++){
            docPair=ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
                    if (valAra[docPair]<min){
                        min=valAra[docPair];
                    }
                }
            }
        }
    }
    return min;
}
```

```
double findMaxSel(int [ ]valAra, int num){
//returns integer maximum value in valAra of those documents in selection set
    double max=-999;
    int docPair;
    for (int dA=0;dA<num;dA++){
        for (int dB=dA+1;dB<num;dB++){
            docPair=ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
                    if (valAra[docPair]>max){
                        max=valAra[docPair];
                    }
                }
            }
        }
    }
    return max;
}

double findMinUnSelPairs(double [ ] diff, boolean [ ] selPossible,  int numDocs,
                                                    int minCommon, int [ ] crAra){
//find value of avg differnce of those documents which are still possible for selection
```

```
    double minVal=statMax;
    int doc=0;
    for (int docA=0;docA<numDocs;docA++){
        for (int docB=docA+1;docB<numDocs;docB++){
            doc = ndxPair[docA][docB];
            if (doc>=0){
                if ((selPossible[docA] || selPossible[docB]) && (diff[doc]< minVal) &&
                                   (crAra[doc]>=minCommon)){
                    minVal = diff[doc];
                }
            }
        }
    }
    if (minVal > (statMax-tol))  {
        System.out.println("****Error in findMinUnSel  minumum reset to "+minVal);
        System.out.println("****Program will end");
        System.exit(0);
    }
    return minVal;
}


boolean moreSel(boolean [ ] selectP, int numDocs){
```

```
//finds if there are more documents to be considered for selection
    boolean retVal = false;
    int doc=0;
    while(doc<numDocs&&!retVal){
        if (selectP[doc] == true){
            retVal=true;
        }
        doc++;
    }
    return retVal;
}

void calcAvgSelDiff(boolean [ ] valid, double [ ] avgSelDiff,boolean [ ] select,
                                                            int numDocs ) {
//calculates the average difference for those documents currently selected
    double [ ] sum = new double[numDocs];
    int [ ] num = new int[numDocs];
    int docPair;
    for (int docA=0;docA<numDocs;docA++){
        sum[docA] = 0;
        num[docA]=0;
        for (int docB=0;docB<numDocs;docB++){
            docPair = ndxPair[docA][docB];
            if (docPair>=0){
                if (select [docA] && select[docB] && docA != docB){
                    sum[docA] = sum[docA]+normDiff[docPair];
                    num[docA]++;
                }
            }
        }
    }
    for (int docA=0;docA<numDocs;docA++){
        if (num[docA]>0)
            avgSelDiff[docA] = sum[docA]/num[docA];
        else
            avgSelDiff[docA]=statMax;
    }
}

void addMinDocs(int numDocs, boolean [ ]valid, boolean [ ] select,
                                        boolean [ ] selPossible, int minCommon, int[ ]
crAra){
//atempt to add those docs from pairs that have the miniumum normalized difference
//if none are eligible, add docs with minimum avgDiff
    double min = findMinUnSelPairs(normDiff,selPossible,numDocs,minCommon,
crAra);
```

```
boolean docsAdded = false;
int docPair=0;
for (int docA=0;docA<numDocs;docA++){
    for (int docB=docA+1;docB<numDocs;docB++){
        if (valid[docA] && valid[docB]) {
            docPair = ndxPair[docA][docB];
            if ((docPair>=0) && (selPossible[docA] || selPossible[docB])){
                if ((normDiff[docPair] <=  min) &&(crAra[docPair]>=minCommon ))  {
                    if (selPossible[docA]){
                        docsAdded = true;
                        select[docA] = true;
                        selPossible[docA] = false;
                    }
                    if (selPossible[docB]){
                        docsAdded = true;
                        select[docB] = true;
                        selPossible[docB] = false;
                    }
                }
            }
        }
    }
}
```

```
    if (!docsAdded) {
        double minVal=statMax;
        for (int d=0; d<numDocs; d++) {
            if ((avgDiff[d] < minVal) && selPossible[d])
                minVal = avgDiff[d];
        }
        for (int d=0; d<numDocs; d++) {
            if ((avgDiff[d] <= minVal) && selPossible[d]){
                docsAdded = true;
                selPossible[d] = false;
                select[d]=true;
            }
        }
    }
    if (!docsAdded) {
        System.out.println("Error  --- selection possible but no docs added");
        System.exit(0);
    }
}

void selectBestDocs(double [ ]ratioAra, int [ ] valAra, boolean [ ] select,
                                        int numDocs, double thresh,boolean [ ] valid, int caseNo,
int numDeSels,
                                        int minCommon) {
//selects those documents that have the minimum difference in the set of unselected
    documents and deselects documents whose difference exceeds the threshhold
    int deSelect[ ] = new int[numDocs];
    double avgSelDiff[ ] = new double[numDocs];
    boolean selPossible[ ] = new boolean[numDocs];
    for (int doc=0;doc<numDocs;doc++){
        selPossible[doc]=valid[doc];
        select[doc] = false;
        deSelect[doc] = 0;
    }
    ckPossible(crAra,valid,selPossible,minCommon,numDocs);
    calcNormDiff(ratioAra, valAra, numDocs );
    calcAvgDiff(valid, avgDiff, numDocs);
    int docPair=0;
    for (int docA = 0; docA < numDocs; docA++){
        for (int docB = docA+1; docB<numDocs; docB++){
            docPair = ndxPair[docA][docB];
        }
    }
    double min=statMax;
    while(moreSel(selPossible,numDocs)){
        addMinDocs(numDocs, valid, select, selPossible, minCommon,crAra);
```

```
        calcAvgSelDiff(valid, avgSelDiff, select, numDocs );
        for (int d=0;d<numDocs;d++){
            if (select[d] && (avgSelDiff[d] > thresh)){
                select[d] = false;
                deSelect[d]++;
                if (deSelect[d] < numDeSels) {
                    selPossible[d] = true;
                }
            }
        }
        ckPossible(crAra,valid,selPossible,minCommon,numDocs);
        calcAvgSelDiff(valid,avgSelDiff,select, numDocs );
    }
    int selCnt=0;
    for (int docA=0;docA<numDocs;docA++){
        for (int docB=docA+1;docB<numDocs;docB++){
            int dPair = ndxPair[docA] [docB];
                if (dPair>=0){
                    selectPair[dPair] = true;
                if (select[docA]==false || select[docB]==false)
                    selectPair[dPair]=false;
                    if (selectPair[dPair]){
                        selCnt++;
                }
            }
        }
    }
    if (selCnt<2){
        System.out.println("***Error--less than 2 document Pairs selected.  " +
                                            "\nProgram will       exit. CaseNo="+caseNo);
        System.out.println("***Rerun with different parameters");
        System.exit(0);
    }
}

boolean checkMeasures(boolean h, int numDocs){
//flags runs where any measure is invalid
//noted by negative value when measurements calculated
    boolean measBad=false;
    int docPair;
    for (int docA = 0; docA < numDocs; docA++){
        for (int docB = docA+1; docB<numDocs; docB++){
            docPair = ndxPair[docA][docB];
            if (docPair>=0){
                    if (simAra[docPair]<0)
                    measBad=true;
                if (jacAra[docPair]<0)
                    measBad=true;
                if (vecAra[docPair]<0)
                    measBad=true;
                if (h) {
                    if (simInfAra[docPair]<0)
                        measBad=true;
                    if (jacInfAra[docPair]<0)
                        measBad=true;
                    if (vecInfAra[docPair]<0)
                        measBad=true;
                    if (gcsmAra[docPair]<0)
                        measBad=true;
                    if (uhiAra[docPair]<0)
                        measBad=true;
                    if (whiAra[docPair]<0)
                            measBad=true;
                }
            }
        }
    }
    return measBad;
}
```

```
boolean checkVariance(boolean h, int num, int caseNo) {
//flags runs where there is no variance
//in the data for a specific measurements
    double low, hi;
    boolean varBad=false;
    low=findMinSel(simAra,num);
    hi=findMaxSel(simAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
    low=findMinSel(jacAra,num);
    hi=findMaxSel(jacAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
    low=findMinSel(vecAra,num);
    hi=findMaxSel(vecAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
    if (caseNo<2){
        low=findMinSel(crAra,num);
        hi=findMaxSel(crAra,num);
        if (Math.abs(low-hi)<tol)
            varBad=true;
```

```
}
else{
    low=findMinSel(crAra,num);
    hi=findMaxSel(crAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
}
if (h){
    low=findMinSel(simInfAra,num);
    hi=findMaxSel(simInfAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
    low=findMinSel(jacInfAra,num);
    hi=findMaxSel(jacInfAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
    low=findMinSel(vecInfAra,num);
    hi=findMaxSel(vecInfAra,num);
    if (Math.abs(low-hi)<tol)
        varBad=true;
    low=findMinSel(gcsmAra,num);
    hi=findMaxSel(gcsmAra,num);
    if (Math.abs(low-hi)<tol)
```

```
            varBad=true;
        low=findMinSel(uhiAra,num);
        hi=findMaxSel(uhiAra,num);
        if (Math.abs(low-hi)<tol)
            varBad=true;
        low=findMinSel(whiAra,num);
        hi=findMaxSel(whiAra,num);
        if (Math.abs(low-hi)<tol)
            varBad=true;
    }
    return varBad;
}


void validate(boolean hierarchy, int numDocs,  int caseNo) {
//end run if any measurement was invalid or if all measurements equal
    boolean badRun=false;
    badRun=checkMeasures(hierarchy, numDocs);
    if (!badRun)
        badRun=checkVariance(hierarchy, numDocs,caseNo);
        if (badRun){
        System.out.println("***Invalid measurements or no variance within measure in
case”
                                                            +caseNo);
        System.out.println("***Rerun with different parameters");
        System.exit(0);
    }
}

void ckPossible(int[ ] crAra, boolean[ ] valid, boolean [ ]selPossible,
                                    int minCommon, int numDocs){
// see if there is any combination for this document that is possible
    int numPossible=0;
    for (int docA=0;docA<numDocs;docA++){
        if (selPossible[docA])
            numPossible++;
    }
    if (numPossible>1){
        int docPair=0;
        for (int docA=0;docA<numDocs;docA++){
            boolean posPair=false;
            if (selPossible[docA]){
                for (int docB=0;docB<numDocs;docB++){
                    if (selPossible[docB]){
                        docPair=ndxPair[docA][docB];
                        if (docPair >= 0){
                            if (crAra[docPair] >= minCommon) {
```

```
                        posPair = true;
                    }
                }
            }
        }
    }
    selPossible[docA]=posPair;
  }
 }
}

void calcNormDiff( double[ ] ratioAra,  int[ ] valAra, int numDocs){
//calculates the normalized difference between the ratio array and the value array
// that is used to estimate the precision value for the document pair
    int docPair=0;
    double normCommon[ ] = new double[numUnique];
    double normRatio[ ] = new double[numUnique];
    double maxRatio = findMax(numDocs,ratioAra);
    double maxCommon = (double)findMax(numDocs,valAra);
    for (int docA=0;docA<numDocs;docA++){
        for (int docB=docA+1;docB<numDocs;docB++){
            docPair=ndxPair[docA][docB];
            if (docPair>=0){
                normRatio[docPair] = ratioAra[docPair]/maxRatio;
                normCommon[docPair] = valAra[docPair]/maxCommon;
                normDiff[docPair] = normRatio[docPair]-normCommon[docPair];
                    if (normDiff[docPair] < 0)
                        normDiff[docPair] = -1*normDiff[docPair];
            }
        }
    }
}

void calcAvgDiff(boolean [] valid, double[ ] avgDiff, int numDocs ){
//calculates the average difference of each document for all the pairs
//using that document in the selection set
    double [ ] sum = new double[numDocs];
    int [ ] num = new int[numDocs];
    int docPair;
    for (int docA=0;docA<numDocs;docA++){
        sum[docA] = 0;
        num[docA]=0;
            for (int docB=0;docB<numDocs;docB++){
                docPair = ndxPair[docA][docB];
                if (docPair>=0){
                        if (docA != docB) {
```

```
                        sum[docA] = sum[docA]+normDiff[docPair];
                        num[docA]++;
                    }
                }
            }
        }
        for (int docA=0;docA<numDocs;docA++){
            if (num[docA]>0){
                avgDiff[docA] = sum[docA]/num[docA];
            }
            else { avgDiff[docA]=statMax;
        }
    }
}

double findMax(int num, double [] ara) {
//returns the maximum value in the array of n elements of type double
    double maxV = -1 * statMax;
    int docPair=0;
    for (int dA=0; dA<num; dA++) {
        for (int dB=dA+1;dB<num;dB++){
            docPair = ndxPair[dA][dB];
            if (docPair >= 0){
                if (ara[docPair] > (maxV-tol)){
                    maxV = ara[docPair];
                }
            }
        }
    }
    return maxV;
}


int findMax(int num, int [] ara){
//returns the maximum value in the array of n elements of type integer
    int maxV = -1 * statMax;
    int docPair=0;
    for (int dA=0; dA<num; dA++) {
        for (int dB=dA+1;dB<num;dB++){
            docPair = ndxPair[dA][dB];
            if (docPair >= 0){
                if (ara[docPair] > maxV){
                    maxV = ara[docPair];
                }
            }
        }
    }
    return maxV;
}

voidcalcCorr(intcaseNo,intnum,boolean[]sel){
//calculates the appropriate correlation value based on case number
//for each of the various simulation measures
    if (caseNo<2){
        simCc = findCorCoef(crAra,simAra,num);
        simInfCc = findCorCoef(crAra,simInfAra,num);
        jacCc = findCorCoef(crAra,jacAra,num);
        jacInfCc = findCorCoef(crAra,jacInfAra,num);
        vecCc = findCorCoef(crAra,vecAra,num);
        vecInfCc = findCorCoef(crAra,vecInfAra,num);
        gcsmCc = findCorCoef(crAra,gcsmAra,num);
        uhiCc = findCorCoef(crAra,uhiAra,num);
        whiCc = findCorCoef(crAra,whiAra,num);
        simRCc = findRankCorCoef(rankCr,simRankAra,num,sel);
        simInfRCc = findRankCorCoef(rankCr,simInfRankAra,num,sel);
        jacRCc = findRankCorCoef(rankCr,jacRankAra,num,sel);
        jacInfRCc = findRankCorCoef(rankCr,jacInfRankAra,num,sel);
        vecRCc = findRankCorCoef(rankCr,vecRankAra,num,sel);
        vecInfRCc = findRankCorCoef(rankCr,vecInfRankAra,num,sel);
        gcsmRCc = findRankCorCoef(rankCr,gcsmRankAra,num,sel);
        uhiRCc = findRankCorCoef(rankCr,uhiRankAra,num,sel);
        whiRCc = findRankCorCoef(rankCr,whiRankAra,num,sel);
    }
    else{
        simCc = findCorCoef(curAra,simAra,num);
        simInfCc = findCorCoef(curAra,simInfAra,num);
        jacCc = findCorCoef(curAra,jacAra,num);
        jacInfCc = findCorCoef(curAra,jacInfAra,num);
        vecCc = findCorCoef(curAra,vecAra,num);
        vecInfCc = findCorCoef(curAra,vecInfAra,num);
        gcsmCc = findCorCoef(curAra,gcsmAra,num);
        uhiCc = findCorCoef(curAra,uhiAra,num);
        whiCc = findCorCoef(curAra,whiAra,num);
        simRCc = findRankCorCoef(rankCUR,simRankAra,num,sel);
        simInfRCc = findRankCorCoef(rankCUR,simInfRankAra,num,sel);
        jacRCc = findRankCorCoef(rankCUR,jacRankAra,num,sel);
        jacInfRCc = findRankCorCoef(rankCUR,jacInfRankAra,num,sel);
        vecRCc = findRankCorCoef(rankCUR,vecRankAra,num,sel);
        vecInfRCc = findRankCorCoef(rankCUR,vecInfRankAra,num,sel);
        gcsmRCc = findRankCorCoef(rankCUR,gcsmRankAra,num,sel);
        uhiRCc = findRankCorCoef(rankCUR,uhiRankAra,num,sel);
        whiRCc = findRankCorCoef(rankCUR,whiRankAra,num,sel);
    }
```

```
}


double findCorCoef(int [ ] x, double [ ] y, int num ) {
//calculates the correlation coefficient (Pearson's correlation coeffcient)
    double r;
    double minY=findMinSel(y,num);
    double maxY=findMaxSel(y,num);
    if (Math.abs(minY-maxY)<tol)
        r=-99.0;
    else{
        double sumX = 0;
        double sumXsq = 0;
        double sumY = 0;
        double sumYsq = 0;
        double sumXY = 0;
        double numSel=0;
        int docPair=0;
        for (int dA=0; dA<num; dA++){
            for (int dB=dA+1; dB<num; dB++) {
                docPair=ndxPair[dA][dB];
                if (docPair>=0){
                    if (selectPair[docPair]){
                        sumX += x[docPair];
                    sumXsq += x[docPair]*x[docPair];
                    sumY += y[docPair];
                    sumYsq += y[docPair]*y[docPair];
                    sumXY += x[docPair]*y[docPair];
                    numSel++;
                }
            }
        }
    }
    double numer = (numSel*sumXY) - (sumX*sumY);
    double termX1 = numSel*sumXsq;
    double xVal=termX1-(sumX*sumX);
    double termY1 = numSel*sumYsq;
    double yVal=termY1-(sumY*sumY);
    double den =Math.sqrt(xVal*yVal);
    if (den>0){
        r = numer/den;
    }
    else
        r=-99;
  }
  return r;
}
```

```
double findRankCorCoef (double [ ] ara1, double [ ] ara2, int num, boolean[ ]sel) {
//calculates the ranked correlation coefficient (Spearman's correlation coeffcient)
    double rankCor;
    double min2=findMinSel (ara2,num);
    double max2=findMaxSel (ara2,num);
    if (Math.abs(min2-max2)<tol)
        rankCor=-99.0;
    else{
        double diffSq = 0;
        double diff = 0;
        double numS=0;
        int docPair=0;
        double sumDiffSq=0;
        for (int dA=0;dA<num;dA++){
            for (int dB=dA+1;dB<num;dB++){
                docPair=ndxPair[dA][dB];
                if (docPair>=0){
                    if (sel[dA]&&sel[dB]&&selectPair[docPair]){
                        diff = ara1[docPair]-ara2[docPair];
                        diffSq = diff*diff;
                        sumDiffSq = sumDiffSq + diffSq;
                        numS=numS+1;
                    }
                }
            }
        }
        double denom = (numS*numS*numS)-numS;
        if (denom>0)
            rankCor = 1 - ((6.0*sumDiffSq)/denom);
        else{
            rankCor=-99;
            System.out.println("***Error--Too few documents selected to “+
            ”calculate ranked correlation  numS="+numS);
            System.out.println("***Rerun with different parameters.");
            System.exit(0);
        }
    }
    return rankCor;
}

int findCountIn1stDocNot2nd(int dA, int dB, Annotations annot,int numConcepts,
                                                                        boolean inferred){
//counts the number of document pairs that  have a specific concept annotated
//(or inferred, based on boolean flag)in the first document but not both
    int count = 0;
    if (!inferred){
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc] ==1)
&&(annot.docAnnotMatrix[dB][tc]!=1)){
                count++;
            }
        }
    }
    else{
        for (int tc=0;tc<numConcepts;tc++) {
            if (((annot.docAnnotMatrix[dA][tc] ==1) ||(annot.docAnnotMatrix[dA][tc]
==2))
                    &&((annot.docAnnotMatrix[dB][tc]<1) ||
(annot.docAnnotMatrix[dB][tc]>2))){
                count++;
            }
        }
    }
    return count;
}

int findCountIn2ndDocNot1st(int dA, int dB, Annotations annot, int numConcepts,
                                                    boolean inferred){
```

```
//counts the number of document pairs that  have a specific concept annotated
//(or inferred, ased on boolean flag)in the second document but not both
    int count = 0;
    if (!inferred){
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc] !=1)
&&(annot.docAnnotMatrix[dB][tc]==1)){
                count++;
            }
        }
    }
    else{
        for (int tc=0;tc<numConcepts;tc++) {
            if (((annot.docAnnotMatrix[dB][tc] ==1) ||(annot.docAnnotMatrix[dB][tc] ==2))
                        &&((annot.docAnnotMatrix[dA][tc]<1) ||
(annot.docAnnotMatrix[dA][tc]>2))){
                count++;
            }
        }
    }
    return count;
}
```

```
int findCountInNeither(int dA, int dB, Annotations annot, int numConcepts,
                                                    boolean inferred){
//counts the number of document pairs that do not have a specific concept annotated
//(or inferred, based on boolean flag)in either document
   int count = 0;
   if (!inferred){
      for (int tc=0;tc<numConcepts;tc++) {
         if ((annot.docAnnotMatrix[dA][tc] !=1) &&
   (annot.docAnnotMatrix[dB][tc]!=1)){
            count++;
         }
      }
   }
   else{
      for (int tc=0;tc<numConcepts;tc++) {
         if (((annot.docAnnotMatrix[dA][tc] < 1) || (annot.docAnnotMatrix[dA][tc] > 2))
&&
                     ((annot.docAnnotMatrix[dB][tc] < 1) ||
(annot.docAnnotMatrix[dB][tc]> 2))){
            count++;
         }
      }
   }
```

```
    return count;
}


int findCountInEither(int dA, int dB, Annotations annot, int numConcepts,
                                                boolean inferred){
//counts the number of document pairs that  have a specific concept annotated
//(or inferred, based on boolean flag)in the either document
    int count = 0;
    if (!inferred){
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc] ==1) ||
(annot.docAnnotMatrix[dB][tc]==1)){
                count++;
            }
        }
    }
    else{
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc] ==1) ||(annot.docAnnotMatrix[dA][tc] ==2)||
                        (annot.docAnnotMatrix[dB][tc]==1) ||
(annot.docAnnotMatrix[dB][tc]==2)){
                count++;
            }
        }
    }
    return count;
}

int findCountInBoth (int dA, int dB, Annotations annot, int numConcepts,
                                                            boolean inferred){
//counts the number of document pairs that  have a specific concept annotated
//(or inferred, based on boolean flag)in the both documents
    int count = 0;
    if (!inferred){
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc] ==1)
&&(annot.docAnnotMatrix[dB][tc]==1)){
                count++;
            }
        }
    }
    else {
        for (int tc=0;tc<numConcepts;tc++) {
            if (((annot.docAnnotMatrix[dA][tc] ==1)|| (annot.docAnnotMatrix[dA][tc]
==2))
```

```
            && ((annot.docAnnotMatrix[dB][tc]==1) || (annot.docAnnotMatrix[dB][tc]
==2))){
                count++;
            }
        }
    }
    return count;
}

int findParentCountInEither(int dA, int dB, Annotations annot, Ontology onto){
//counts the number of document pairs that have the parent
//of a specific concept annotated (or inferred, based on boolean flag) in either document
    int count = 0;
    int [ ] parentAAra = new int [onto.conceptCt];
    int [ ] parentBAra = new int [onto.conceptCt];
    for (int tc=0;tc<onto.conceptCt;tc++) {
        parentAAra[tc] = 0;
        parentBAra[tc] = 0;
    }
    for (int tc=0;tc<onto.conceptCt;tc++) {
        for (int p1=0;p1<onto.concepts.maxLowestParents;p1++) {
            if ((annot.docAnnotMatrix[dA][tc]==1) &&
                                        (onto.concepts.lowestParent[tc][p1]>=0))
                parentAAra[onto.concepts.lowestParent[tc][p1]] = 1;
            if ((annot.docAnnotMatrix[dB][tc] ==1) &&
                                        (onto.concepts.lowestParent[tc][p1]>=0))
                parentBAra[onto.concepts.lowestParent[tc][p1]] = 1;
    }
    if ((parentAAra[tc] > 0) || (parentBAra[tc] > 0))
        count++;
    }
return count;
}

int findParentCountInBoth(int dA, int dB, Annotations annot, Ontology onto){
//counts the number of document pairs that have the parent
//of a specific concept annotated or inferred, based on boolean flag) in both documents
    int count = 0;
    int [ ] parentAAra = new int [onto.conceptCt];
    int [ ] parentBAra = new int [onto.conceptCt];
    for (int tc=0;tc<onto.conceptCt;tc++) {
        parentAAra[tc] = 0;
        parentBAra[tc] = 0;
    }
    for (int tc=0;tc<onto.conceptCt;tc++) {
        for (int p1=0;p1<onto.concepts.maxLowestParents;p1++) {
            if ((annot.docAnnotMatrix[dA][tc]==1) &&
                                            (onto.concepts.lowestParent[tc][p1]>=0))
                parentAAra[onto.concepts.lowestParent[tc][p1]] = 1;
            if ((annot.docAnnotMatrix[dB][tc] ==1) &&
                                            (onto.concepts.lowestParent[tc][p1]>=0))
                parentBAra[onto.concepts.lowestParent[tc][p1]] = 1;
        }
        if ((parentAAra[tc] >0) && (parentBAra[tc] >   0))
            count++;
    }
    return count;
}

double findDotProd (int dA, int dB, Annotations annot, int numConcepts,
                                                boolean inferred) {
//finds the dot product of the vectors representing documents docA and docB
    int annotA = 0;
    int annotB = 0;
    double sum = 0;
    if (!inferred){
        for (int tc=0;tc<numConcepts;tc++) {
            if (annot.docAnnotMatrix[dA][tc]==1)
                annotA = 1;
            else
                annotA = 0;
            if (annot.docAnnotMatrix[dB][tc]==1)
                annotB = 1;
            else
                annotB=0;
            sum = sum + (annotA * annotB);
        }
    }
    else{
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc]==1) || (annot.docAnnotMatrix[dA][tc]==2))
                annotA = 1;
            else
                annotA = 0;
            if ((annot.docAnnotMatrix[dB][tc]==1) || (annot.docAnnotMatrix[dB][tc]==2))
                annotB = 1;
            else
                annotB=0;
            sum = sum + (annotA * annotB);
        }
    }
    return sum;
```

```
}


double findPDotProd(int dA, int dB, Annotations annot, Ancestors ancestors) {
//finds the dot product of the parent vectors
//if there is more than one parent for any annotated concept, the concepet
//is considered to have a valid parent annotation if any of the parents are annotated
    int numParents = ancestors.numParents;
    int annotA=0;
    int annotB=0;
    double sum = 0;
    for (int pc =0;pc<numParents;pc++){
        if (annot.docParentMatrix[dA][pc]==1)
            annotA=1;
        else
            annotA=0;
        if (annot.docParentMatrix[dB][pc]==1)
                annotB=1;
        else
            annotB=0;
        sum=sum+(annotA*annotB);
    }
    return sum;
}
```

```
double findWpDotProd (int dA, int dB, Annotations annot, Ancestors ancestors,
                                                    int numConcepts) {
//finds weighted dot product of the parent vectors
//weights of the parent vectors were calcuated in the Annotations method
   int parentNdx = 0;
   int parentNum = 0;
   int np=ancestors.numParents;
   double annotA = 0;
   double annotB = 0;
   double sumn = 0;
   double sumsqA = 0;
   double sumsqB=0;
   for (int pc=0;pc<np;pc++) {
      annotA = (double)    annot.countParentMatrix[dA][pc] /
                           (double)(annot.countAnnotsInDoc[dA]);
      sumsqA=sumsqA+annotA*annotA;
      annotB = (double) annot.countParentMatrix[dB][pc] /
                           (double) (annot.countAnnotsInDoc[dB]);
      sumsqB=sumsqB+annotB*annotB;
      sumn = sumn + (annotA * annotB);
   }
return sumn;
```

```
}


double findSqrtSumSqr(int dA, Annotations annot, int numConcepts, boolean inferred) {
//calculates the square root of the sum values in the annotation vector
//equivalent to the sum of the squares of the values, since all values are either 0 or 1
    int annotValA;
    double sum = 0;
    if (!inferred){
        for (int tc=0;tc<numConcepts;tc++) {
            if (annot.docAnnotMatrix[dA][tc] == 1)
                annotValA = 1;
            else
                annotValA = 0;
            sum += annotValA;
        }
    }
    else {
        for (int tc=0;tc<numConcepts;tc++) {
            if ((annot.docAnnotMatrix[dA][tc] == 1)||(annot.docAnnotMatrix[dA][tc] == 1))
                annotValA = 1;
            else
                annotValA = 0;
            sum += annotValA;
        }
    }
    sum = Math.sqrt(sum);
    return sum;
}

double findPSqrtSumSqr(int dA, Annotations annot, Ancestors ancestors) {
//calculates the square root of the sum values of the parent nodes of the annotated terms
//equivalent to the sum of the squares of the values, since all values are either 0 or 1
    int num = ancestors.numParents;
    int annotValA;
    double sum = 0;
    for (int tc=0;tc<num;tc++) {
        if (annot.docParentMatrix[dA][tc] == 1)
            annotValA = 1;
        else
            annotValA = 0;
        sum += (annotValA* annotValA);
    }
    sum = Math.sqrt(sum);
    return sum;
}
```

```
double findWpSqrtSumSqr(int dA, Annotations annot, Ancestors ancestors) {
//calculates the weighted square root of the sum values
//of the parent nodes of the annotated terms
//equivalent to the sum of the squares of the values, since all values are either 0 or 1
    int num = ancestors.numParents;
    double annotValA;
    double sum = 0;
    for (int tc=0;tc<num;tc++) {
        annotValA =(double) annot.countParentMatrix[dA][tc] /
                                    (double)(annot.countAnnotsInDoc[dA]);
        sum = sum + (annotValA*annotValA);
    }
    sum = Math.sqrt(sum);
    return sum;
}


double calcGcsm(int docA, int docB, Annotations annot, Ancestors ancestors, Ontology
onto){
// calculates the gcsm similarity measure
    double retval = 0;
    for (int termA = 0; termA <onto.conceptCt;termA++) {
        if (annot.docAnnotMatrix[docA] [termA] ==1) {
            for (int termB = 0; termB <onto.conceptCt; termB++){
                if (annot.docAnnotMatrix[docB] [termB]==1) {
                    retval+=ancestors.lVector[termA][termB];
                }
            }
        }
    }
    return retval;
}


void calcRanks(int caseNo, int numDocs, Annotations annot) {
//calls methods to rank results of similarity analysis based on number of commmon roots,
    common unexpected roots and the various similarity measures
    int num=numUnique;
    calcRanking(rankCr,crAra,numDocs,annot);
    calcRanking(rankCUR,curAra,numDocs,annot);
    calcSelRank(simRankAra, simAra,numDocs);
    calcSelRank(simInfRankAra, simInfAra,numDocs);
    calcSelRank(jacRankAra, jacAra,numDocs);
    calcSelRank(jacInfRankAra, jacInfAra,numDocs);
    calcSelRank(vecRankAra, vecAra,numDocs);
    calcSelRank(vecInfRankAra, vecInfAra,numDocs);
    calcSelRank(gcsmRankAra, gcsmAra,numDocs);
    calcSelRank(uhiRankAra, uhiAra,numDocs);
    calcSelRank(whiRankAra, whiAra,numDocs);
}


void calcRanking(double [ ] rankAra, int [ ] valAra, int num, Annotations annot){
//calculates the rankings of values in a value array
//and stores those values in the rank array
//in case of duplicate values, the average of the rankings is used
    int numThisRank = 0;
    int prevMin =-1;
    double currentRank = 1;
    int min = -999;
    int docPair=0;
    min = findMin(prevMin, valAra,num);
    while (min < statMax){
        numThisRank =0;
        for (int dA=0; dA<num; dA++){
            for (int dB=dA+1; dB < num; dB++){
                docPair = ndxPair[dA][dB];
                if (docPair>=0){
                    if (selectPair[docPair]){
                        if (valAra[docPair]== min) {
                            rankAra[docPair] = currentRank;
                            numThisRank++;
                        }
                    }
                }
            }
        }
        if (numThisRank>1)
            currentRank =  resetRank(rankAra,(int)currentRank,numThisRank, num);
        else
            currentRank++;
        prevMin=min;
        min=findMin(prevMin, valAra,num);
    }
    for (int dA=0; dA<num; dA++){
        for (int dB=dA+1; dB < num; dB++){
            docPair = ndxPair[dA][dB];
            if (docPair>=0){
                if (selectPair[docPair]){
                    if (valAra[docPair]== min) {
                        rankAra[docPair] = currentRank;
                        numThisRank++;
                    }
                }
            }
        }
    }
}

void calcSelRank(double [ ] rankAra, double [ ] valAra,  int num){
//calculates rank of values from those documents in selection set
    int selCt=0;
    int numThisRank = 0;
    double prevMin = -1;
    int currentRank = 1;
    double min = -999;
    int docPair = 0;
    while (rankMoreSel(rankAra,num)) {
    min = findMin(prevMin, valAra, num);
    numThisRank =0;
    for (int dA=0; dA<num; dA++){
        for (int dB=dA+1; dB<num; dB++){
            docPair=ndxPair[dA][dB];
            if (docPair>=0){
                if (selectPair[docPair]){
                    if ((valAra[docPair] >= min-tol) &&   (valAra[docPair] <= min+tol)) {
                        rankAra[docPair] = currentRank;
                        numThisRank++;
                    }
                }
            }
        }
    }
    if (numThisRank>1)
        currentRank =  resetRank(rankAra, currentRank, numThisRank, num);
    else
        currentRank++;
    prevMin=min;
    }
}


int resetRank(double [ ] rankAra, int cur, int freq, int num){
//assigns an average value to all the rankings that are equal and resets the next rank value
    double hi = freq +cur -1;
    double rankNew = (cur+hi)/2.0;
    int docPair=0;
    for (int dA=0; dA<num; dA++){
        for (int dB=dA+1; dB<num; dB++){
            docPair = ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
```

```
                    if ((rankAra[docPair] > cur-tol) && (rankAra[docPair] < cur+tol)){
                        rankAra[docPair]=rankNew;
                    }
                }
            }
        }
    }
    return (int)(hi+1);
}


void initRanks(int caseNo){
//calls appropriate methods to initialize the ranking arrays
    if (caseNo<2){
        initSelRanks(rankCr);
    }
    else{
        initSelRanks(rankCUR);
    }
    initSelRanks(simRankAra);
    initSelRanks(simInfRankAra);
    initSelRanks(jacRankAra);
    initSelRanks(jacInfRankAra);
    initSelRanks(vecRankAra);
    initSelRanks(vecInfRankAra);
    initSelRanks(gcsmRankAra);
    initSelRanks(uhiRankAra);
    initSelRanks(whiRankAra);
}

void initSelRanks(double [ ]  rankAra) {
//initializes rankAra to negative value
    for (int doc = 0; doc < numUnique; doc++){
        rankAra[doc] = -999;
    }
}


boolean rankMoreSel(double [ ] ara, int num) {
//returns a value to indicate if there are more values to be ranked
    int docPair=0;
    boolean retVal = false;
    for (int dA=0; dA<num; dA++){
        for (int dB=0; dB<num; dB++){
            docPair = ndxPair[dA][dB];
            if (docPair>=0){
                if (selectPair[docPair]){
                    if (ara[docPair] <-1 && selectPair[docPair]) {
                        retVal=true;
                    }
                }
            }
        }
    }
    return retVal;
}

int findMin(int prevMin, int [ ] ara, int num){
//method finds the integer minimum value in the array
//which exceeds the previous minimum and returns that value
    int min = statMax;
    int docPair=0;
    for (int dA=0; dA<num; dA++){
        for (int dB=dA+1; dB<num; dB++){
            docPair = ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
                    if (ara[docPair] < min && ara[docPair] > prevMin){
                        min = ara[docPair];
                    }
                }
            }
        }
    }
    return min;
}


double findMin(double prevMin, double [ ] ara,int num){
//method finds the double minimum value in the array
//which exceeds the previous minimum and returns that value
    double startMin = 999;
    double min=startMin;
    int docPair=0;
    for (int dA=0; dA<num; dA++){
        for (int dB=dA+1;dB<num;dB++){
            docPair = ndxPair[dA][dB];
            if (docPair >=0){
                if (selectPair[docPair]){
                    if (ara[docPair] < min && ara[docPair] > (prevMin+tol)) {
                        min = ara[docPair];
                    }
                }
            }
        }
    }
    if (min>=startMin)
        min=prevMin;
    return min;
}


void writeFile(String runID, int docUsed,  Annotations annot,
                                        TermDocumentMatrix tdm, String docID, int caseNo,
double [] thresh){
//writes the output data results to a file that can be opened later
//by a spreadsheet application and also print the results to the screen
    runID=runID+" Thresh"+thresh[caseNo];
    File output = new File(docID+runID);
    if (output.exists())
        output.delete();
    try{
        BufferedWriter out = new BufferedWriter(new FileWriter(output.getPath(),true));
        int finalSel=0;
        out.write("ThisRunID = " +runID+"\t\t\tfor " + docID + "\n");
        out.write("\nDoc1\tDoc2"+
                                "\tCommonRoots"+
                                "\tRankCommonRoots"+
                              "\tUnexpectedCommonRoots"+
                              "\tRankUnexpectedCommonRoots"+
                              "\tEither"+"\tBoth"+
                              "\tSameParentEither"+"\tSameParentBoth"+


"\tSimMatch\tRankSimMatch\tSimInfMatch\tRankSimInfMatch"+
                              "\tJMatch\tRankJMatch\tJInfMatch\tRankJInfMatch"+
                              "\tCosine\tRankCosine\tCosineInf\tRankCosineInf"+
                              "\tGcsm\tRankGcsm\tUhi\tRankUhi\tWhi\tRankWhi\n");
  System.out.print("ThisRunID = " +runID+"\tfor " + docID + "\n");
  boolean [ ]rootFound = new boolean[tdm.numRoots];
  boolean [ ]annotFound = new boolean[tdm.numRoots];
  int dCtPair=0;
  for (int d1 = 0; d1<docUsed;d1++){
    for (int d2 = d1+1;d2<docUsed;d2++){
      dCtPair=ndxPair[d1][d2];
        if (dCtPair>=0){
          if (selectPair[dCtPair]){
          finalSel++;
          out.write(d1 + "\t " + d2+ "\t"+
          crAra[dCtPair] + "\t"+ rankCr[dCtPair] + "\t" + curAra[dCtPair]   +
          "\t"+rankCUR[dCtPair]+"\t"+
          either[dCtPair] +"\t"+both[dCtPair]+"\t"+
                        numSameParentEither[dCtPair]
+"\t"+numSameParentBoth[dCtPair]+"\t"+
                        simAra[dCtPair] + "\t" +
                        simRankAra[dCtPair] + "\t" +
                        simInfAra[dCtPair] + "\t" +
                        simInfRankAra[dCtPair] + "\t" +
                        jacAra[dCtPair] + "\t" +
                        jacRankAra[dCtPair] + "\t" +
                        jacInfAra[dCtPair] + "\t" +
                        jacInfRankAra[dCtPair] + "\t" +
                        vecAra[dCtPair] + "\t" +
                        vecRankAra[dCtPair] + "\t" +
                        vecInfAra[dCtPair] + "\t" +
                        vecInfRankAra[dCtPair] + "\t" +
                        gcsmAra[dCtPair] + "\t" +
                        gcsmRankAra[dCtPair] + "\t" +
                        uhiAra[dCtPair] + "\t" +
                        uhiRankAra[dCtPair] + "\t" +
                        whiAra[dCtPair] + "\t" +
                        whiRankAra[dCtPair]+"\n");
                      }
                    }
                  }
```

```
    }
    out.write("Corr.\t\t\t\t\t\t\t\t\t\t"+simCc+"\t\t"+simInfCc+"\t\t"+jacCc+"\t\t"+


jacInfCc+"\t\t"+vecCc+"\t\t"+vecInfCc+"\t\t"+gcsmCc+"\t\t"+uhiCc+"\t\t"+
                        whiCc+"\n");
    out.write("Rank Corr.\t\t\t\t\t\t\t\t\t\t\t"+simRCc+"\t\t"+simInfRCc+"\t\t"+jacRCc+


"\t\t"+jacInfRCc+"\t\t"+vecRCc+"\t\t"+vecInfRCc+"\t\t"+gcsmRCc+"\t\t"+
                        uhiRCc+"\t\t"+whiRCc+"\n");
    out.write("Number of pairs selected=\t"+finalSel);
    System.out.print("Corr.\t\t\t\t\t\t\t\t\t\t"+simCc+"\t\t"+simInfCc+"\t\t"+


jacCc+"\t\t"+jacInfCc+"\t\t"+vecCc+"\t\t"+vecInfCc+"\t\t"+gcsmCc+"\t\t"+
                          uhiCc+"\t\t"+whiCc+"\n");
    System.out.print("Rank Corr.\t\t\t\t\t\t\t\t\t\t\t"+simRCc+"\t\t"+simInfRCc+"\t\t"+


jacRCc+"\t\t"+jacInfRCc+"\t\t"+vecRCc+"\t\t"+vecInfRCc+"\t\t"+
                          gcsmRCc+"\t\t"+uhiRCc+"\t\t"+whiRCc+"\n");
    System.out.println("Number of pairs selected=\t"+finalSel);
    out.close();
}
catch(Exception e){
    System.err.println("Error writing File Output in stats");
```

```
        System.err.println("Program will end");
        System.err.println("Error is " + e);
        e.printStackTrace( );
        System.exit(1);
    }
}
}


class Stop{
//This class models the stop list used to eliminate common non-descriptive words
String [ ] stopList = {
"a","about","above","across","after","again","all","almost","alone","along","already",
"also","although","always","an","and","another","any","anybody","anyone","anything",
"anywhere","are","around","as","ask","asked","asking","asks","at","away",
"b","back","backed","backing","backs","be","because","become","becomes","became",
"been","before","began","begin","begun","behind","being","beings","best","better",
"big","both","but","by",
"c","came","can","cannot","can't","case","cases","certain", "certainly",
"clear","clearly","come","could",
"d","did","differ","different","differently","do","does","doesn't",
"done","down","downed","downing","during",
"e","each","early","either","end","ended","ending","ends","enough",
```

"even","evenly","ever", "every","everybody","everyone","everything","everywhere",

"f","face","faces","fact","facts","far","felt","few","find","finds",

"for","four","from","full","fully","further","furthered","furthering","furthers",

"g","gave","get","gets","give","given","gives", "go","goes","going",

"good","goods","got", "great","greater","greatest",

"h","had", "has","have","having","he","her","herself","here","high","higher","highest",

"him","himself","his","how","however",

"i","if", "in","into","is","it","its","it's","itself",

"j","just",

"k","keep","keeps","kind","knew","know","known","knows",

"l","large","largely","later","latest","least","less",

"let","lets","let's","like","likely","long","longer","longest",

"m","made","make","making","man","many","may","me","member", "members","men",

"might","more","most","mostly","mr","mrs","much","must","my","myself",

"n","necesary","need","needed","needing","needs","never","next","no","non",

"none","not","nobody","noone","nothing","now","nowhere","number","numbers",

"o","of","off","old","older","oldest","on","once","one",

"only","open","opened","opening","opens","or","other","others","our", "out",

"p","part","parted","parting", "parts","per","perhaps","place","places",

"point","pointed","pointing","points","possible","put","puts",

"q","quite",

"r","rather","really","right","room","rooms",

"s","said","same","saw","say","says","see","sees","seem","seemed","seeming","seems",

```
"several","shall","she","should","show","showed","showing","shows","side","sides",
"since","small","smaller","smallest","so","some","somebody","someone",
“something","somewhere",
"still","such","sure",
"t","take","taken","than","that","the","their","them","then","there","therefore",
"these","they","thing","things","think",
“thinks","this","those","though","thought","thoughts","three","through","thus","to",
"today","together","too","took","toward","turn","turned","turning","turns","two",
"u","under","until","up","upon","us","use","used","uses",
"v","very",
"w","want","wanted","wanting","wants","was","way","ways","we","well","wells",
"went","were","what","when","where","whether","which","while","who",
"whole","whose","why","will","with","within","without","would",
"y","year","years","yet","you","young","younger","youngest","your","yours",
"z"};

Stop( ){ }

void elimStops(Words w){
//eliminates words in stop list and single character words
    boolean [] temp = new boolean[w.wordCt];
    String [ ] tempWord = new String[w.wordCt];
    for (int ct=0; ct<w.wordCt; ct++) {
```

```
    temp[ct]=true;
    tempWord[ct] = w.theWords[ct];
}
boolean done=false;
int wc;
int lastStop=0;
boolean stopWord=false;
try{
    for (wc=0; wc<w.wordCt; wc++){
    stopWord=false;
    int ndxStop=lastStop;
        while (ndxStop<stopList.length && !stopWord){
        stopWord = ((w.theWords[wc].equalsIgnoreCase(stopList[ndxStop])) ||
                                        (w.theWords[wc].length() <2));
            if (stopWord){
                temp[wc] = false;
                lastStop=ndxStop;
            }
            ndxStop++;
        }
    }
    int ndx=0;
    for (int ct=0;ct<w.wordCt;ct++){
            if (temp[ct]) {
                w.theWords[ndx] = tempWord[ct];
                ndx++;
            }
        }
        w.wordCt=ndx;
    }
    catch (Exception e) {
        System.err.println("\nerror in stop " + e);
        e.printStackTrace();
        System.exit(1);
    }
}
}
import java.io.*;
class TermDocumentMatrix{
//This class models the matrix used to store results
//depicting which terms are present in documents
String [ ] [ ] words;
int [ ] numWords;
boolean [ ] termsUsed;
int [ ] ndxNextTerm;
String [ ] termList;
```

```
int [ ]  [ ] termDocGrid;

int [ ] rootIndex;

int numTerms;

String [ ] roots;

int [ ] [ ] rootInDocument;

int numRoots;

int [ ] rootInHowManyDocs;


TermDocumentMatrix( ){

    System.err.println("***Warning Default TermDocumentMatrix Constructed" +

                                                   " with no number of docs");

    System.err.println("***Program will exit");

    System.exit(1);

}


TermDocumentMatrix(int nDocs, int maxWords){

    words = new String [nDocs] [maxWords];

    numWords = new int[nDocs];

    termsUsed = new boolean[nDocs];

    ndxNextTerm = new int[nDocs];

    for (int nd =0; nd < nDocs; nd++){

        numWords[nd] = 0;

        termsUsed[nd] = false;
        ndxNextTerm[nd] = 0;
        for (int nw = 0; nw<maxWords; nw++){
            words[nd] [nw] = "";
        }
    }
}

void createMaster(Annotations annot, int nDocs) {
//creates the master term / document matrix
    numTerms=0;
    for (int nd=0;nd<nDocs;nd++){
        numTerms=numTerms+numWords[nd];
    }
    termDocGrid = new int [numTerms] [nDocs] ;
    termList = new String [numTerms];
    rootIndex = new int [numTerms];
    roots = new String [numTerms];
    rootInDocument = new int [nDocs][numTerms] ;
    rootInHowManyDocs = new int[numTerms];
    numRoots = 0;
    for (int nt = 0; nt < numTerms; nt++) {
        for (int nd = 0; nd < nDocs; nd++){
            rootInDocument[nd] [nt] = 0;
            termDocGrid[nt] [nd] = 0;
        }
        rootInHowManyDocs[nt] = 0;
    }
    Words w = new Words(numTerms);
    int wc=0;
    for (int nd=0;nd<nDocs;nd++){
        for (int nw=0; nw<numWords[nd];nw++){
            w.theWords[wc] = words[nd][nw];
            wc++;
        }
    }
    w.sortWords(0,numTerms-1);
    w.elimDups( );
    numTerms = w.wordCt;
    termList = w.theWords;
    for (int nt=0;nt<numTerms;nt++){
        for (int nd=0; nd < nDocs; nd++){
            if (termInDoc(termList[nt],nd)){
                termDocGrid[nt] [nd]= 1;
            }
        }
    }
```

```
    for (int nt=0;nt<numTerms;nt++){
        for (int nd=0; nd < nDocs; nd++){
            if (termInDoc(termList[nt],nd)){
                termDocGrid[nt] [nd]= 1;
            }
        }
    }
}

void addDoc(int docNum, Words wt){
//stores the words in the document text for document number docNum
    numWords[docNum] = wt.wordCt;
    words[docNum] = wt.theWords;
}

boolean termInDoc (String term, int nd) {
//determines if term is a part of the text of document nd
    boolean found = false;
    int tn=0;
    while (tn<numWords[nd] && !found) {
        if (words[nd][tn].equalsIgnoreCase(term))
            found=true;
        tn++;
    }
    return found;
}


void findRoots(String runID, Annotations annot, int nDocs) {
//finds the roots of all terms in the documents
    int numRoots = 0;
    String root;
    boolean found = false;
    for (int term = 0; term  < numTerms; term++) {
        root=termList[term];
        updateRootList(root, term, nDocs);
    }
    int rootNdx=0;
    for (int term = 0; term  < numTerms; term++) {
        rootNdx = rootIndex[term];
        int documentCt = 0;
        for (int docNum=0; docNum < nDocs; docNum++){
            if (termDocGrid[term][docNum] > 0)
            documentCt ++;
            rootInHowManyDocs[rootNdx] =

    rootInHowManyDocs[rootNdx] + documentCt;
```

```
        }
    }
}

void updateRootList(String root, int termNum, int nDocs){
// updates the rootList information for a specific root
    boolean found = false;
    int cnt = 0;
    int thisRoot = 0;
    while (cnt < numRoots && !found){
        if (roots[cnt].equalsIgnoreCase(root)) {
            found= true;
            thisRoot = cnt;
        }
        else
            cnt++;
    }
    if (!found) {
        roots[numRoots] = root;
        thisRoot = numRoots;
        numRoots++;
    }
    rootIndex[termNum] = thisRoot;
    for (int docNum = 0; docNum < nDocs; docNum++){
        if (termDocGrid[termNum][docNum] >0) {
            rootInDocument[docNum] [thisRoot] =1;
        }
    }
}


void assignStatus(Annotations annot, Ontology onto, int nDocs){
//assigns a status of 0 to terms not present in a document,
//1 if term is expected from annnotation,
//2 if term is inferred from annotations (part of superConcept)
//3 if term is unexpected
    for (int nd = 0; nd < nDocs; nd++) {
        int status=0;
        if (annot.validSheet[nd]){
            for (int nt = 0; nt < numTerms; nt++){
                if (rootInDocument[nd][rootIndex[nt]] > 0)
                    status = annot.findStatus(nt,nd,onto,roots[rootIndex[nt]]);
                else
                    status=0;
                rootInDocument[nd][rootIndex[nt]]=status;
            }
        }
    }
}
}

class Words {
//This class models the words for sections of text
String [ ] theWords;
int wordCt;

Words(){};

Words (int numwords){
    theWords = new String[numwords];
    wordCt=numwords;
}

void printWords ( ) {
//prints words in object --used for checkout
    System.out.println("wordCt"+ wordCt);
    for (int num=0; num<wordCt; num++)
        System.out.print("\t"+theWords[num]);
    System.out.println(" ");
}
```

```
void stripWords(){
//eliminate leading or trailing special characters in words
//all characters after apostrophe will be eliminated
    for (int nw=0; nw<wordCt;nw++){
        String word = theWords[nw];
        char first, last;
        char apos = '\'';
        boolean hasApos = false;
        for (int ct=1;ct<word.length();ct++){
            if (word.charAt(ct) == apos) {
                word=word.substring(0,ct);
            }
        }
        first = word.charAt(0);
        if (word.length() > 0){
            while (strip(first,word)){
                word = word.substring(1);
                    first = word.charAt(0);
            }
        }
        last = first;
        if (word.length() > 0) {
            last = word.charAt(word.length()-1);
            while (strip (last,word)) {
                word=word.substring(0,word.length()-1);
                last=word.charAt(word.length()-1);
            }
            theWords[nw] = word;
        }
    }
}

boolean strip (char ch, String s){
//returns TRUE value if ch is special character
//or if ch has ANSII value less than 0
    char [ ] special = {' ','<','>',',','.',':','"',';','[',']','"',
    '[',']','`','~','!','@','#','$','%','^','&',
    '*','(',')','-','_','+','=','/','\\','\''};
    int len = special.length;
    if (s.length()<=1)
        return false;
    else {
        for (int ct=0; ct<len; ct++){
            if (ch==special[ct])
                return true;
        }
    }
    if (Character.getNumericValue(ch)<0)
        return true;
    return false;
}

void sortWords(int start, int end) {
//performs a quick sort of data in theWords array
    if (start >= end)
        return;
    int ptn = split (start, end);
    sortWords (start, ptn);
    sortWords (ptn+1,end);
}

int split(int start, int end)  {
//partitions the words array based and returns the pivot element
    String pivot = theWords[start];
    int low = start-1;
    int high = end + 1;
    while (low < high) {
        low++;
        while (theWords[low].compareTo(pivot)<0)
            low++;
        high--;
        while (theWords[high].compareTo(pivot)>0)
            high--;
        if (low < high)
            swapWords (low, high);
    }
    return high;
}

void swapWords(int low, int high){
//swaps theWords[low] and theWords[high]
    String temp= theWords[low];
    theWords[low] = theWords[high];
    theWords[high]= temp;
}

void elimDups( ){
//eliminates duplicate words in theWords
    for (int wc =0; wc < wordCt-1; wc++) {
        if (theWords[wc].equals(theWords[wc+1])){
            for (int nw=wc+1;nw<wordCt-1;nw++)
            theWords[nw]=theWords[nw+1];
          wc--;
          wordCt--;
        }
    }
}

void findRoots(Roots textRoot){
//stores words from text root object in words object
    for (int wc =0; wc < wordCt-1; wc++) {
        theWords[wc]=textRoot.findRoot(theWords[wc]);
    }
}
}
```

Reference List

Abiteboul, S., Buneman, P., & Suciu, D. (2000). *Data on the Web From Relations to Semistructured Data and XML*. San Francisco: Morgan Kaufmann Publishers.

Agrawal, R. & Srikant, R. (1994). Fast algorithms for mining data association rules. *Proceedings of the 20th International Conference on Very Large Databases.* 487-499.

Anyanwu, K., Maduko, A., & Sheth, S. (2005). SemRank: ranking complexity relationship search results on the Semantic Web. *Proceedings of the 14th International World Wide Web Conference.* 117-127.

Beckett, D., Miller, E., & Brickley, D. (2002). Expressing simple Dublin Core in RDF/XML. Retrieved from http://dublincore.org/documents/2002/07/31/dcmes-xml/ Jan 21, 2003.

Benetti, I., Beneventano, D., Bergamaschi, S., Guerra, F., & Maurizio, V. (2002) An information integration framework for e-commerce. *IEEE Intelligent Systems*, 17, (1). 18-25.

Bot, R. & Wu, Y. (2004). Improving document representations using relevance feedback: the RFA algorithm. *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*. 270-278.

Brickley, D., Buswell, S., Matthews, B., Miller, L., Reynolds, D., & Wilson, M. (2002). SWAD-Europe: Semantic Web Advanced Development in Europe. *Proceedings of the First International Semantic Web Conference.* 409-413.

Brickley, D. & Guha, R. (Ed.) (2003). RDF Vocabulary Description Language 1.0: RDF SchemaW3C Working Draft 23 January 2003. W3C Working Draft 23 January, 2003. Retrieved February 6, 2003 from http://www.w3.org/TR/rdf-schema/.

Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., & Horrocks, I. (2001). Enabling knowledge representation on the Web by extending RDF schema. *Proceedings of the Tenth International World Wide Web Conference on the World Wide Web*. 467-478.

Buneman, P. (1997). Semistructured data. *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* 117-121.

Chakrabarti, S. (2003). *Mining the Web Discovering Knowledge from Hypertext Data*. San Francisco: Morgan Kaufmann Publishers.

Chen, H. & Karger, D. (2006). Less is More: Probabilistic Models for Retrieving Fewer Relevant Documents in an Information Retrieval System. Retrieved March 19, 2005 from http://people.csail.mit.edu/harr/papers/sigir2006.pdf

Dublin Core Metadata Initiative. (2002). Retrieved January 15, 2003 from http://www.dublincore.org

Ehrig, M. & Sure, Y. (2004). Ontology mapping – an integrated approach. *The Semantic Web:Research and Applications. Proceedings of the First European Semantic Web Symposium 2004.* 76-89.

Fensel, D. [ed]. (2000). The semantic web and its languages. *IEEE Intelligent Systems,* 15(6), 67-73.

Fox, C. (1990). A stop list for general text. *SIGIR Forum*. 24(12), 19-35.

Ganesan, P., Garcia-Molina, H., & Widom, J. (2003). Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems, 21, (1).* 64-93.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., Vassalos, V., & Widom, J. (1997). The TSIMMIS approach to mediation: data models and languages. *IEEE Intelligent Systems*, 8(2), 117-132.

Goldman, R. (2000). Integrated query and search of databases, XML and the web. *Digital Dissertations.* (UMI No. 9986100).

Haustein, S. & Pleumann, J. (2002). Is participating in the Semantic Web too Difficult? *Proceedings of the first International Semantic Web Conference*. 448-453.

Haveliwala, T., Gionis, A., Klein, D., & Indyk, P. (2003). Evaluating strategies for similarity search on the web. *Proceedings of the Eleventh International Conference on World Wide Web.* 432-441.

Heflin, J. (2001). Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment. *Digital Dissertations.* (UMI No. 3024929).

Heflin, J. (Ed.) (2003). Web Ontology Language Use Cases and Requirements. W3C Working Draft 3 February, 2003. Retrieved February 6, 2003 from http://www.w3.org/TR/2003/WD-webont-req-20030203.

International Semantic Web Conference 2002 Web Page. (2002). Retrieved July 23, 2002 from http://iswc.semanticweb.org.

Kifer, M., Lausen, G., & Wu, J. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the Association for Computing Machinery*, 42(4), 741-843.

Li, G., Oren, V., Motta, E., Shum, S., & Domingue, J. (2002). ClaiMaker: Weaving a Semantic Web of Research Papers. *Proceedings of the First International Semantic Web Conference.* 436-441.

Lin, D. & Pantel, P. (2001). Induction of semantic classes from natural language text. *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 317*-322.

Lin, S.-H., Shih, C-S., Chen, M., Ho, J.-M., Ko, M.-T., & Huang, Y.-M. (1998). Extracting classification knowledge of Internet documents with mining term associations: a semantic approach. *Proceedings of the 21st Annual International ACM SIGR Conference on Research and Development in Information Retrieval.* 241-249.

Liu, B., Ma, Y., & Yu, P. (2001). Discovering unexpected information from your competitors' web sites. *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 144-153.

Lopatenko, A. (2001). Information retrieval in current research information systems. *Position paper at Workshop on Knowledge Markup and Semantic Annotation at K-Cap'2001*. Retrieved July 5, 2002 from http://semannot2001.aifb.uni-karlsruhe.de/positionpapers/Lopatenko.pdf

Losee, R. (1998). *Text Retrieval and Filtering: Analytic Models of Performance.* Boston: Kluwer Academic Publishers.

Lowry, R. (Ed.) (2003). Concepts and Applications of Inferential Statistics. Retrieved December 19, 2007 from http://faculty.vassar.edu/lowry/webtext.html.

Maedche, A. (2002). *Ontology Learning for the Semantic Web.* Boston: Kluwer Academic Publishers.

Maedche, A. & Staab, S. (2000). SEAL – Discovering conceptual relationships from text. *Proceedings of ECAI-2000.,* Amsterdam: IOS Press, 2002. Retrieved from http://www.aifb.uni-karlsruhe.de/WBS/publications/2000/aifb400_amaetal_2000.pdf.

Maedche, A., Staab, S., Studer, R., Sure, Y., & Volz, R. (2000). SEAL – Tying up information integration and web site management by ontologies. *IEEE-CS Data Engineering Bulletin, Special Issue on Organizing and Discovering the Semantic Web,* March, 2002.

McGuiness, D., Fikes, R., Rice, J., & Wilder, J. (2000). An environment for merging and testing large ontologies. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*. 132-146.

Object Management Group. (2005). Retrieved October 23, 2005 from http://www.omg.org.

O'Reilly, T. (2005). What is Web 2.0? Retrieved August 15, 2007 from http://oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

Papakonstantinou, Y., Garcia-Monila, H., & Widom, J. (1995). Object exchange across heterogeneous information sources. *Proceedings of the IEEE International Conference on Data Engineering.* 251-260.

Pluempitiwiriyawej, C. (2001). A new hierarchical clustering model for speeding up the reconciliation of XML-based data in meditation systems. *Digital Dissertations.* (UMI No. 3925123)

Protégé Project. (2002). Welcome to the Protégé Project. Retrieved May 15, 2002 from http://protege.stanford.edu.

Rasmussen, S. (1992). *An Introduction to Statistics with Data Analysis.* Belmont, CA. Wadsworth, Inc. 513-552.

Salton, G. & Buckley, C. (1988). Term-weight approaches in automatic retrieval. *Information Processing and Management.* 24(5): 513-523.

Salton, G. & McGill, M. (1983). *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.

Sintek, M. & Decker, S. (2002). TRIPLE – A Query, Inference, and Transformation Language for the Semantic Web. *Proceedings of the First International Semantic Web Conference.* 364-378.

Sure, Y. (2001). SWRC The Semantic Web Research Community Ontology. Retrieved July 31, 2002 from http://ontobroker.semanticweb.org/ontos/swrc.html.

Sure, Y., Erdman, M., Angele, J., Staab, S., Studer, R., & Wenke, D. (2002). OntoEdit:Collaborative Ontology Development for the Semantic Web. *Proceedings of the First International Semantic Web Conference.* 221-235.

Vaschillo, A. (2000). A semantic paradigm for intelligent data access. *Digital Dissertations.* (UMI No. 9966108).

W3C (2003). Retrieved February 8, 2003 from http://www.w3c.org.

W3C (2008). Retrieved January 10, 2008 from http://www.w3c.org/2001/sw/SW-FAQ

Wang, K. & Liu, H. (2000). Discovering structural association of semistructured data. *IEEE Transactions on Knowledge and Data Engineering, 12, (3).* 353-371.

Welcome to Ontoknowledge (2002). Retrieved December 15, 2002 from http://www.ontoknowledge.org.

Yang, Y. & Liu, X. (1999). A re-examination of text categorization methods. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information* Retrieval. 42-49.

Zobel, J. & Moffat, A. (1998). Exploring the Similarity Space. *ACM SIGIR Forum,* 32, (1). 18-34.