2012

Developing a Semantic Web Crawler to Locate OWL Documents

Ronald Dean Koron
Wright State University

Developing a Semantic Web Crawler to Locate OWL Documents

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

By

RONALD DEAN KORON
B.S.C.S., University of South Florida, 1984

2012
Wright State University

WRIGHT STATE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

September 6, 2012

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Ronald Dean Koron ENTITLED Semantic Web Crawling with Reasoning Support BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Pascal Hitzler, Ph. D
Thesis Director

Mateen Rizki, Ph. D
Department Chair

Committee on
Final Examination

Pascal Hitzler, Ph. D

Gouzhu Dong, Ph. D

Krishnaprasad Thirunarayan, Ph. D

Andrew T. Hsu, Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Koron, Ronald. M.S. Department of Computer Science and Engineering, Wright State University, 2012. Developing a Semantic Web Crawler to Locate OWL Documents.

The terms **Semantic Web** and **OWL** are relatively new and growing concepts in the World Wide Web. Because these concepts are so new there are relatively few applications and/or tools for utilizing the potential power of this new concept. Although there are many components to the Semantic Web, this thesis will focus on the research question, "How do we go about developing a web crawler for the Semantic Web that locates and retrieves OWL documents." Specifically for this thesis, we hypothesize that by giving URIs to OWL documents, including all URIs from within these OWL documents, priority over other types of references, then we will locate more OWL documents than by any other type of traversal. We reason that OWL documents have proportionally more references to other OWL documents than non-OWL documents do, so that by giving them priority we should have located more OWL files when the crawl terminates, than by any other traversal method.

In order to develop such an OWL priority queue, we needed to develop some heuristics to *predict* OWL documents during real-time parsing of Semantic Web documents. These heuristics are based on filename extensions and OWL language constructs, which are not absolute when predicting a document type before retrieval. However, if our reasoning is correct, then URIs found in an OWL document will likely lead to more OWL documents, such that when the crawl ends because of reaching a maximum document limit, we will have retrieved more OWL documents than by other methods such as breadth-first or load-balanced. We conclude our research with an evaluation of our results to test the validity of our hypothesis and to see if it is worthy of future research.

Contents

List of Tables

ACKNOWLEDGEMENTS

I would like to thank **Prof. Dr. Pascal Hitzler** for introducing me to the world of Semantic Web crawling and for giving me the opportunity to work on this thesis. His patience, guidance, and unrelenting support enabled me to accomplish this effort.

In addition, I would like to give a special thanks to **Ph.D. student Kunal Sengupta**. Without his unwavering support and patience with my Java coding skills (or lack thereof), I would not have been able to pull this off. I greatly appreciate all the time and effort that he gave me that went above and beyond what was required.

Finally, I would like to give a heartfelt "thank you" to my loving and beautiful wife of 28 years, **Holly**. Her understanding and encouragement is what kept me going through this long and arduous process.

Chapter 1

Introduction

One of the current movements in the fields of computer science and specifically, the area knowledge representation and reasoning, is the move toward the "Semantic Web." Ten years ago, the term "Semantic Web" was just a seemingly pie-in-the-sky concept coined by Tim Berners-Lee [Wright 1997], whom many regard as the inventor of the World Wide Web (aka "the Web"). Well, that pie-in-the-sky concept turned into an initiative of the World Wide Web Consortium (W3C) to make information on the World Wide Web readable not only by humans but also by computers. From this initiative came the birth of the still emerging Semantic Web, which will become a major component of the Web 3.0, and thus bring that pie-in-the-sky down to earth.

Although there are many components to the Semantic Web, this thesis will focus on the research question, "How do we go about developing a web crawler to locate Web Ontology Language (OWL) documents on the Semantic Web." This web crawler will traverse, or "crawl," anywhere on the Web, from web document to web document by following each Uniform Resource Identifier (URI) found in documents that have already been parsed. URIs can be comprised of a Uniform Resource Name (URN) or a Uniform Resource Locator (URL) or both. For obvious reasons, the web crawler will only follow URLs, but we will refer to them as URIs because many documents on the Semantic Web contain references to URNs as well. The crawl is kicked off from seed URIs that are retrieved and parsed first. Specifically for this thesis, we hypothesize that by giving URIs to OWL documents, including all URIs from within these OWL documents, priority over other types of references, then we will locate more OWL documents than by any other type of traversal. We reason that OWL

documents have proportionally more references to other OWL documents than non-OWL documents do, so that by giving them priority we should have located more OWL files when the crawl terminates (by number of rounds or documents), than by any other traversal method.

1.1 Ontology

Before we begin discussing the background for this thesis, specifically the Web *Ontology* Language, let us introduce the concept of ontology. Ontology is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related. The current consensus for the definition of ontology was given by Thomas Gruber, from the Knowledge System Laboratory at Stanford University, when he said, "An ontology is an explicit specification of a conceptualization." [Gruber 1993] What this means is that an ontology defines the terms used to describe and represent the subject matter, or domain, of knowledge. Ontologies can be used by people, databases, and applications that need to share domain information.

In 2009, Jeff Heflin stated in *Web Ontology Language (OWL) Use Cases and Requirements* that,

> Ontologies figure prominently in the emerging Semantic Web as a way of representing the semantics of documents and enabling the semantics to be used by web applications and intelligent software agents. Ontologies can prove very useful as a way of structuring and defining the meaning of the resources and properties that are currently being collected and subsequently standardized. Using ontologies, Semantic Web applications can be "intelligent," in the sense that they can work closer to the human conceptual level. [Heflin 2004]

1.2 Thesis Overview

The rest of this thesis is comprised of Chapters 2 through 7, detailing the research and work that went into the research question, and the resulting conclusions to our hypothesis. Chapter 2, titled simply "Background," gives us a much needed background into the software

technologies needed for this research. Chapter 3, titled "Crawling the Semantic Web," gets us started into the development of a web crawler for the Semantic Web. For this research, we decided to build upon an already existing open source web crawler, rather than develop one from scratch. Chapter 4, titled "Enhancing the Crawler," chronicles the enhancements made to the chosen web crawler to provide the capabilities needed for our research. Chapter 5, titled "Crawler Evaluation," details the findings of our OWL document finding crawler, and evaluates our hypothesis. Chapter 6, titled "Future Research," covers future research that could spawn from this research. Finally, Chapter 7, titled "Conclusion," provides a final conclusion to this thesis.

Chapter 2

Background

Because the subject of this thesis is on the forefront of development of the World Wide Web (aka "the Web"), it is imperative that we delve into the software technologies discussed within this research. This will pave the way by providing the essential background information needed to understand and appreciate this research.

This chapter begins with a brief introduction into the Semantic Web concept, followed by sections describing the Extensible Markup Language (XML), the Resource Description Framework (RDF), the RDF Schema (RDFS), and the Web Ontology Language (OWL) languages. Next, we cover RDF Triple Stores, specifically the Virtuoso Universal Server, the SPARQL Protocol and RDF Query Language (SPARQL) used to access and manipulate the RDF Triple Stores, and the web of "Linked Data" that will be used for crawling with the Semantic Web Crawler. We conclude this chapter with a section describing the Eclipse Integrated Development Environment (IDE) that was used for this research.

2.1 The Semantic Web

In the past two decades the Web has evolved conceptually right before our very eyes. Basically, the Web is nothing more than a diverse system, standardized by the World Wide Web Consortium (W3C), of interlinked hypertext documents that can be accessed via the Internet. What began as a novelty has grown into an ubiquitous and essential tool of the modern world.

The Web was born in the late 1990's and was comprised mostly of static webpages

created by content creators known as "webmasters." These webpages were used mostly for the display of static content to consumers in search of such content. These consumers could only view the content but were not allowed to reflect or comment on the content of the webpages. Sure, there were webpages that allowed users to interact using canned forms and scripted buttons, but the information displayed was predefined and the interactions were relatively simplex in nature. This stage of evolution was coined "Web 1.0."

Around the turn of the 21st century, the Web started to grow and evolve into what is now referred to as "Web 2.0" (aka "the Social Web"). This stage of evolution introduced the term "social media," where users can interact and collaborate with each other, are allowed to reflect and comment on the content of webpages, and post content in many forms. Graham Cormode and Balachander Krishnamurthy summarize it when they state "the essential difference between Web 1.0 and Web 2.0 is that content creators were few in Web 1.0 with the vast majority of users simply acting as consumers of content, while any participant can be a content creator in Web 2.0 and numerous technological aids have been created to maximize the potential for content creation." [Cormode and Krishnamurthy 2008]

This leads us to the next stage of evolution, referred to as "Web 3.0," and one of its main components, the "Semantic Web." The next version of the Web will provide a web of linked data that expands upon the concept of presenting human readable documents to the user. The Semantic Web will provide meaning, or semantics, to data such that computers can understand, analyze, and respond to information provided by a website. Instead of content providers structuring just the "look and feel" of a web document, they will be able to provide knowledge that can be used by computers to deductively reason with in order to infer even more knowledge. Tim Berners-Lee, regarded as the inventor of the World Wide Web, put it best we he said "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [Berners-Lee et al. 2001] This thesis focuses on how to develop a web crawler program to crawl this Semantic Web.

2.2 XML

The first step in the direction of the Semantic Web was the emergence of XML as the standard for structuring and exchanging data over the Web. This was a step in the right direction, but because it is only semi-structured, it does not meet the criteria needed for the Semantic Web. At this time, there are a large number of Web services that can be queried for a wealth of high quality information. However, answers to these queries are returned in the semi-structured format of XML where we cannot directly access nor observe the database schema that these services may use internally. Therefore, XML can be used to provide the elementary structure for semantic data, but is not used to model semantics by itself.

2.3 RDF

In 1998 the W3C issued an initial recommendation of a metadata model and language to serve as the basis for an infrastructure of machine-readable semantics, called Resource Description Framework, or RDF. In February 2004, the W3C finalized the RDF recommendation as an official Semantic Web standard. As stated by Claudio Gutiérrez, Carlos Hurtado and Alberto Mendelzon,

> RDF follows the W3C design principles of interpretability, extensibility, evolution and decentralization. In particular, the RDF model was designed with the following goals: simple data model; formal semantics and provable inference; extensible URI-based vocabulary. Achieving these goals allows anyone to make statements about any resource. [Gutiérrez et al. 2004]

As explained in the W3C's *RDF Primer*,

> RDF is a language for representing information and relationships about resources in the Web in terms of uniquely identified attributes and values. In RDF, the universe being modeled is a set of resources identified by Uniform Resource Identifiers, or URIs, and terms of simple properties and property values are used to describe these resources. Much like sentences in human language, RDF statements are comprised of a subject, predicate, and object, comprised of resources (URIs), or string literals.

Both subject and object can be anonymous objects known as a blank nodes. As described in the RDF Primer, RDF models these statements as nodes and arcs in a graph where a statement is represented by:

- a node for the subject.
- a node for the object.
- an arc for the predicate, directed from the subject node to the object node.

Since it is not very convenient to draw graphs when discussing them, an alternative way of writing down the statements, called triples, is used. In the triples notation, each statement in the graph is written as a simple triple of <subject>, <predicate>, and <object>, in that order.

RDF specifies an XML-based syntax (RDF/XML) for recording and exchanging these graphs, which is known as serialization. RDF/XML files can be stored on web servers, retrievable by corresponding URIs. These files are machine processable, and can link bits of information across the Web. However, RDF URIs can also refer to any identifiable thing, including things that may not be directly retrievable on the Web. For instance, a person or a thing, can be referred to by an URI that does not lead to an actual file on the Web. It's just a placeholder to represent that person or thing. Also, RDF properties themselves have URIs, to uniquely identify the relationships between linked resources. [Manola and Miller 2004]

In addition, RDF supports the concepts of (a) entailment for semantics and inference, and (b) reification for describing other RDF statements using RDF. However, we will not delve any further into these concepts because they are out of the scope of this thesis.

2.4 RDFS

While RDF does provide a model to express simple statements about resources and their properties, it is limited in its ability to define the vocabulary used in those statements. There immediately arose a need to provide a language to provide a way of defining specific classes of resources, and the specific property used to describe those resources. From the *RDF Primer*,

> This RDF vocabulary language came to be referred as RDF Schema (RDFS). RDFS does not provide a vocabulary of application-specific classes and properties. Instead, it provides the facilities needed to describe such classes and properties, and to indicate which classes and properties are expected to be used together. The RDFS terms are themselves provided in the form of an RDF vocabulary; that is, as a specialized set of predefined RDF resources with their own special meanings. Therefore, vocabulary descriptions, or schemas, written in the RDFS language are also legal RDF graphs, which can be processed by legacy RDF parsers that are not equipped to handle the RDFS language. [Manola and Miller 2004]

2.5 OWL

The next level above RDF required for the Semantic Web vision is an ontology language that can formally describe the meaning of terminology used in Web documents. If computers are expected to perform useful reasoning and inference tasks on these documents, the language must go beyond the basic semantics of RDFS. That need led to the formal W3C recommendation of OWL 1 in February 2004, at the same time as RDF and RDFS.

2.5.1 OWL 1

From the *OWL Web Ontology Language Overview* we find that,

> OWL 1 extended RDFS by adding more vocabulary for describing properties and classes. These include, among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. This added vocabulary allows users the ability to define and instantiate web ontologies to be used by Semantic Web applications. As with RDFS, ontologies written in the OWL language are also legal RDF graphs and can be represented by a set of RDF triples. [McGuinness and Van Harmelen 2004]

2.5.2 OWL 2

OWL 2, formally recommended in October 2009, is an extension and substantial revision of the OWL 1 language. As specified in the *OWL 2 Web Ontology Language Document Overview* from 2009,

> Like its predecessor, OWL 2 is designed to facilitate ontology development and sharing via the Web, with the ultimate goal of making Semantic Web content more readable to computers. OWL 2 extends OWL 1 by adding new functionality, such as:
>
> - unique key identifiers
> - property chains
> - richer datatypes and data ranges
> - qualified cardinality restrictions
> - asymmetric, reflexive, and disjoint properties
> - enhanced annotation capabilities
>
> OWL 2 also defines three new profiles (OWL 2 Profiles) and a new syntax (OWL 2 Manchester Syntax). In addition, some of the restrictions applicable to OWL DL have been relaxed; as a result, the set of RDF Graphs that can be handled by Description Logics reasoners is slightly larger in OWL 2. [W3C OWL Working Group 2009]

There are two species of OWL 2 in addition to the three new profiles that have been standardized. These are best described by Pascal Hitzler et al. in the *OWL 2 Web Ontology Language Primer*,

> There are two alternative ways of assigning meaning to ontologies in OWL 2 called direct model-theoretic semantics (OWL 2 DL) and the RDF-based semantics (OWL 2 Full). The direct model-theoretic semantics provides a meaning for OWL 2 in a Description Logic style, hence the name OWL 2 DL. Whereas, the RDF-Based Semantics is an extension of the semantics for RDFS and is based on viewing OWL 2 ontologies as RDF graphs, hence the name OWL 2 Full. The difference being that

OWL 2 DL is a syntactically restricted version of OWL 2 Full where the restrictions are designed to make reasoning for implementors. Because OWL 2 Full is considered an undecidable decision problem, there are no reasoners that can be developed to always provide correct "yes or no" answers. However, because of the restrictions to OWL 2 DL, it makes writing a reasoner that, in principle, can return all "yes or no" answers (subject to resource constraints) possible.

In addition to OWL 2 DL and OWL 2 Full, OWL 2 specifies three profiles: (1) OWL 2 EL, (2) OWL 2 QL, and (3) OWL 2 RL. OWL 2, in general, is a very expressive language (both computationally and for users) and thus can be difficult to implement well and to work with. These additional profiles are designed to be approachable subsets of OWL 2 that are sufficient for a variety of applications. As with OWL 2 DL, computational considerations are a major requirement for these profiles. In order to guarantee for scalable reasoning, the existing profiles share some limitations regarding their expressiveness. In general, they disallow negation and disjunction, as these constructs complicate reasoning and have shown to be only rarely needed for modeling. For example, in none of the profiles it is possible to specify that every person is male or female. These profiles will be further discussed in the following subsections. [Hitzler et al. 2009]

2.5.2.1 OWL 2 EL

From the *OWL 2 Web Ontology Language Primer*,

> The EL acronym reflects the profile's basis in the so-called EL family of description logics (EL++); they are languages providing mainly existential quantification of variables. Besides negation and disjunction, OWL 2 EL also disallows universal quantification on properties. Therefore propositions like "all children of a rich person are rich" cannot be stated. Moreover, as all kinds of role inverses are not available, there is no way of specifying that, say, parentOf and childOf are inverses of each other.
>
> Working with OWL 2 EL is fairly similar to working with OWL 2 DL: one can use class expressions on both sides of a subClassOf statement and even infer such

relations. For many large, class-expression oriented ontologies, by only a little simplification one can get an OWL 2 EL ontology and preserve the bulk of the meaning of the original ontology. [Hitzler et al. 2009]

2.5.2.2 OWL 2 QL

Again from the *OWL 2 Web Ontology Language Primer*,

> The QL acronym reflects the fact that query answering in this profile can implemented by rewriting queries into a standard relational Query Language. OWL 2 QL captures many commonly used features in RDFS and small extensions thereof, such as inverse properties and subproperty hierarchies. Among other constructs, OWL 2 QL disallows existential quantification of roles to a class expression, i.e. it can be stated that every person has a parent but not that every person has a female parent. Moreover property chain axioms are not supported. OWL 2 QL restricts class axioms asymmetrically, that is, you can use constructs as the subclass that you cannot use as the superclass.
>
> OWL 2 QL can be realized using standard relational database technology (e.g., SQL) simply by expanding queries in the light of class axioms. This means it can be tightly integrated with RDBMSs and benefit from their robust implementations and multi-user features. Expressively, it can represent key features of Entity-relationship and UML diagrams (at least those with functional restrictions). Thus, it is suitable both for representing database schemas and for integrating them via query rewriting. [Hitzler et al. 2009]

2.5.2.3 OWL 2 RL

Finally, from the *OWL 2 Web Ontology Language Primer*,

> The RL acronym reflects the fact that reasoning in this profile can be implemented using a standard Rule Language. Among other constructs, OWL 2 RL disallows statements where the existence of an individual enforces the existence of another individual: for instance, the statement "every person has a parent" is not expressible

in OWL RL. OWL 2 RL restricts class axioms asymmetrically, that is, you can use constructs as the subclass that you cannot use as the superclass.

The OWL 2 RL profile is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate both OWL 2 applications that can trade the full expressivity of the language for efficiency, and RDF(S) applications that need some added expressivity from OWL 2. This is achieved by defining a syntactic subset of OWL 2 which is amenable to implementation using rule-based technologies, and presenting a partial axiomatization of the OWL 2 RDF-Based Semantics in the form of first-order implications that can be used as the basis for such an implementation.

Suitable rule-based implementations of OWL 2 RL can be used with arbitrary RDF graphs. As a result, OWL 2 RL is ideal for enriching RDF data, especially when the data must be massaged by additional rules. Compared with OWL 2 QL, OWL 2 RL works better when you have already massaged your data into RDF and are working with it as RDF triples. [Hitzler et al. 2009]

2.6 Triple Store

The term "Triple Store" is the common name given to a database management system for RDF Data triples. These systems provide data storage, management and access via APIs, endpoints, and query languages to RDF Data. In reality, many Triple Stores are in fact Quad Stores, due to the need to maintain RDF provenance data within the data management system, but for some reason are still known as Triple Stores. Any Triple Store that supports Named Graph functionality is more than likely a Quad Store.

For this research, we decided to use OpenLink Software's Virtuoso Universal Server[1] as our Triple Store of RDF data acquired during the crawls. The reasons were many; (a) Virtuoso is a native Triple Store, (b) available in an open source edition, (c) compatible with Mac OS X, (d) provides numerous standard data access APIs, (e) supports SPARQL/Update (SPARUL) extension of SPARQL, (f) supports uploading of RDF data to the Triple Store

[1] `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSRDFFAQ`

via SPARQL HyperText Transfer Protocol (HTTP) endpoint, and (g) provides reasoning support. Therefore, this Triple Store can be hosted on my personal computer to store and manage RDF triples and to possibly infer more triples using automatic reasoning.

2.7 SPARQL

According to Eric Prud'hommeaux and Andy Seaborne from their 2008 article *SPARQL Query Language for RDF*,

> Since its official W3C recommendation in January 2008, SPARQL (pronounced "sparkle") is regarded as *the* query language of the Semantic Web. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs. SPARQL was used in our research to access and manage the RDF data stored in our local Virtuoso Triple Store. [Prud'hommeaux and Seaborne 2008]

2.8 Linked Data

The term Linked Data refers to a set of best practices for publishing and linking structured data on the Web. In the words of Christian Bizer, Tom Heath, and Tim Berners-Lee in their 2009 article in the International Journal on Semantic Web and Information Systems, titled *Linked Data – The Story So Far*,

> Linked Data is simply about using the Web to create typed links between data from different sources. These may be as diverse as databases maintained by two organizations in different geographical locations, or simply heterogeneous systems within one organization that, historically, has not easily interoperated at the data level. Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data

sets, and can in turn be linked to form external data sets. While the primary units of the hypertext Web are HTML (HyperText Markup Language) formatted documents connected by untyped hyperlinks, Linked Data relies on documents containing data in RDF format. However, rather than simply connecting these documents, Linked Data uses RDF to make typed statements that link arbitrary things in the world. As we shall see, many of our crawls of the Semantic Web will take us into the Linking Open Data (LOD) community's cloud by nature of the Linked Data. [Bizer et al. 2009]

2.9 Eclipse IDE

Because this research entailed a fair amount Java programming, Eclipse IDE for Java Developers[2] was used for software development and testing. Some of the reasons for selecting Eclipse are: (a) Eclipse IDE is free and open source software, (b) compatiblity with Mac OS X, and (c) its ease of use.This programming environment proved to be invaluable as it saved us time and effort that was needed elsewhere.

[2] `http://www.eclipse.org`

Chapter 3

Crawling the Semantic Web

Before we can start crawling the web looking for OWL documents, we have to implement a web crawler to crawl the Semantic Web first. Due to time constraints and a steep learning curve, we came to the decision that it was best to start with an already developed open source web crawler that we can adapt to meet our needs. We decided that it would be best to find something written in Java or C++ because of availability and my limited programming experience. This way we were able to focus in on the main topic of developing software to identify and find OWL documents.

3.1 Finding an Existing Open Source Web Crawler

Our first question was where to find an open source web crawler. Obviously the answer is to look on utilize Google's web crawlers and search for "open source web crawlers." This gave us many choices. Quite a few could be dismissed immediately because they were not written in Java or C++, or because they didn't fit the mold that we were looking for. However, we did stumble across a few that looked promising. Let's take a look at the prospects to see what they had to offer.

3.1.1 myfocusedcrawler

The myfocusedcrawler[1] web crawler project is hosted on the Google Code website and is licensed under the GNU General Public License (GPL), version 2. This project is comprised

[1] `http://code.google.com/p/myfocusedcrawler`

of a focused crawler framework that implements various focused crawling approaches. It also includes an experimental prototype based on first-order logic rules. This project was considered because it is a "focused" crawler, which has similarities with this research. The downside to this program is that it is written in Perl and it will filter out links to documents that are not part of the focused group instead of crawling to them to see if they do lead to documents within the focused group.

3.1.2 crawler4j

The crawler4j[2] web crawler project is hosted on the Google Code website and is licensed under Apache License version 2.0. It is an open source Java crawler which provides a simple interface for crawling the Web. It provides basic crawling, image, multi-threading, and data/stats collection. The downside to this program is that it is too basic for our needs.

3.1.3 P-Sucker

The P-Sucker[3] program was written in Java and is in the public domain. It is multi-threaded and will crawl the Web, saving all images and video files that are linked. The downside to this program is that it does not adhere to the web crawler politeness policy by checking for a robots.txt file or meta-tags before retrieving a webpage, and that it specializes in image/video files.

3.1.4 JSpider

The JSpider[4] program is 100% pure Java and is developed under the GNU Lesser General Purpose License (LGPL) Open Source license. This program is touted as a highly configurable and customizable web spider engine. However, it has not been updated since 2003, so we would have to bring it up to date.

[2] http://code.google.com/p/crawler4j

[3] http://www.andreas-hess.info/programming/webcrawler/index.html

[4] http://j-spider.sourceforge.net

3.1.5 Heritrix

The Heritrix[5] project is the Internet Archive's open-source web crawler which is licensed under Apache License version 2.0. It is an extensible, web-scalable, archival-quality web crawler program written in Java. The downside to this program is that it stores the web resources it crawls in Arc files, which means that a command-line tool called arcreader has to be used to extract contents from an Arc file.

3.1.6 Nutch

The Apache Nutch[6] system is an open source web-search software project of the Apache Software Foundation which is written in Java and licensed under Apache License version 2.0. This system can be enhanced to parse other document formats by using a highly flexible, easily extensible and thoroughly maintained plugin infrastructure. It is a very robust and highly scalable crawler that is designed for distributed processing on large clusters. In addition, it follows the adhere to the web crawler politeness policy by obeying robots.txt rules. The downside to this program is that it is much more robust than we need for this research.

3.1.7 LDSpider

The LDSpider[7] web crawler project can be downloaded from the Google Code website under the terms of the GNU General Public License (GPL), version 3. As described by Robert Isele et al., in the document *LDSpider: An open-source crawling framework for the Web of Linked Data*,

> Requirements and challenges for crawling the Linked Data web are different from regular web crawling, thus the LDSpider project offers a web crawler adapted to traverse and harvest content from the Linked Data web. LDSpider is an extensible Linked Data crawling framework, enabling client applications to traverse and to consume the Web of Linked Data. The main features of LDSpider are given below:

[5] `https://webarchive.jira.com/wiki/display/Heritrix/Heritrix`
[6] `http://nutch.apache.org/index.html`
[7] `http://code.google.com/p/ldspider`

- It can process a variety of Semantic Web data formats including RDF/XML, Turtle, Notation 3, RDFa and many microformats by providing a plugin architecture to support Any23
  (`http://any23.org`).
- Crawled data can be stored together with provenance meta-data either in a file or via
  SPARQL/UPDATE in an RDF Triple Store.
- It offers different crawling strategies, such as breadth-first and load-balancing traversal, for following RDF links between data items.
- It is usable as a command line application, but also offers a simple API which allows applications to configure and control the details of the crawling process.
- The framework is delivered as a small and compact jar with a minimum of external dependencies.
- The crawler is high-performing by employing a multi-threaded architecture.

LDSpider has been developed in Java to provide a flexible Linked Data crawling framework, which can be customized and extended by client applications. [Isele et al. 2010]

3.2 The Winner Is?

Drumroll please . . . The winner is the obvious choice: LDSpider! As you can see from above, the LDSpider project fits our needs like a glove. It's open source, written in Java, supports SPARQL/UPDATE, provides two different crawling strategies and employs multi-threading. In fact, we could not find any downside to using LDSpider. Once the decision was made, we downloaded the LDSpider project to my home computer and imported it into Eclipse IDE in order to begin making our enhancements.

Chapter 4

Enhancing the Crawler

After we picked LDSpider for our open source web crawler, downloaded the source code, and imported into the Eclipse IDE, it was time to start enhancing it to meet our research needs. First we had to study the code and run some tests with the LDSpider software as is so that we could get an understanding of how it works. Once we felt comfortable with how it worked it was time to start the enhancements.

This chapter chronicles, in a somewhat chronological order, the enhancements made to the LDSpider source code to provide the capabilities needed for our research. We will start with a section that gives a very general description of the core LDSpider implementation. We will not cover every package or class delivered in the original LDSpider package, only the ones pertinent to our research. That is followed by a section describing changes made to provide a list of vocabularies and the reference counts for each of the vocabulary terms found in the crawled documents. Then next section we cover the changes made to allow this program to write RDF data to the Virtuoso Triple Store. Next, we detail the addition of an OWL Priority Queue traversal algorithm. Following that is a section regarding separating the vocabulary statistics in known and unknown vocabularies. We then describe the changes made to add the capability to use a search engine (Google) to provide seed URIs. Finally we discuss adding the parent node and outgoing link count to the document statistics.

4.1 LDSpider Implementation

LDSpider [Isele et al. 2010] is implemented in Java and uses three external libraries. The parsing of RDF/XML, N-Triples and N-Quads is provided by the NxParser[1] library. Because of necessary changes to NxParser code, we downloaded and imported this package into Eclipse IDE also. The HTTP functionality is provided by the Apache HttpClient[2] library, while the web crawler politeness policy is respected through the use of the Norbert[3] library. See Appendix A for a listing of the source files and jar files used for this project. The following subsections will detail the crawling strategies and the URI fetching pipeline components (Java packages).

4.1.1 Crawling Strategies

The order of how LDSpider traverses a graph starting from the given seed URIs is specified by the crawling strategy. The original LDSpider provides two different round-based crawling strategies:

Breadth-First Strategy: Takes three parameters: <depth> <url-limit> <pld-limit>. In each round, LDSpider fetches all URIs extracted from the content of the URIs of the previous round, before advancing to the next round. The depth of the breadth-first traversal, the maximum number of URIs crawled per round and per pay-level domain (domain that requires payment at a TLD or TLD registrar) as well as the maximum number of crawled pay-level domains can be specified. This strategy can be used in situations where only a limited graph around the seed URIs should be retrieved.

Load-Balancing Strategy: Takes a single parameter: <max-urls>. This strategy tries to fetch the specified number of URIs as quickly as possible while adhering to a minimum and maximum delay between two successive requests to the same pay-level domain. The load-balancing strategy is useful in situations where the fetched documents should be distributed between domains without overloading a specific server.

[1] `http://sw.deri.org/2006/08/nxparser`

[2] `http://hc.apache.org`

[3] `http://ww.osjava.org/norbert`

LDSpider will fetch URIs in parallel by employing multiple threads. Also, these strategies can be requested to stay on the domains of the seed URIs.

4.1.2 Fetching Pipeline Components

The core components of LDSpider are considered the "fetching pipeline" because they deal with each URI as though it's going through a pipeline, from one component to the next. The fetching pipeline consists of the following components:

The Frontier: First, the URI is added to the Frontier which holds the URI until the next round of the traversal. This frontier can be ranked by pay-level domain with queues for each pay-level domain, or just added to a set to be ordered later.

The Queue: Once the next round starts, the URI is put into one of a number of queues depending on the traversal strategy. The queues can be ordered alphabetically by pay-level domain (breadth-first), or ordered from largest to smallest sub-queue size, where each sub-queue is assigned a pay-level domain (load balanced). This way the threads can poll from each queue in round-robin fashion so that all the threads are not trying to access the same servers at the same time. If they do, the crawler will then throttle back by delaying 2 seconds before accessing again.

The Fetch Filter: Once a URI is polled from the queue for crawling, the Fetch Filter determines whether a particular page should be fetched by the crawler. This is used to restrict the pages which are crawled to certain MIME types (e.g. to RDF/XML).

The Content Handler: Receives the document and tries to extract RDF data from it. LDSpider includes a content handler for documents formatted in RDF/XML, called NxParser, and a general content handler, which forwards the document to an Any23 server to handle other types of documents including Turtle, Notation 3, RDFa and many microformats. C

The Sink: Receives the extracted statements from the content handler and processes them usually by writing them to some specified output. LDSpider includes sinks for writing various formats including N-Quads and RDF/XML as well as to write directly to a

Triple Store using SPARQL/Update. Both sinks can be configured to write metadata containing the provenance of the extracted statements.

The Link Filter: Receives the parsed statements from the content handler and extracts all URIs which should be fetched in the next round. If determined that they should be fetched, they are added to the Frontier for the next round. In addition, each Link Filter can be configured to restrict crawling to a specific domain.

4.2 Output Statistics

The first thing needed to enhance the original LDSpider was to provide the capability to output and save statistics from the crawls. We began by adding to MAIN another command line option, "-v," with optional arguments for a vocabulary statistics filename and a document statistics filename. If the user does not provide one or both of the filenames, default filenames are provided in CrawlerConstants.java. We then added functionality in the following stages.

4.2.1 List of Vocabularies and Reference Counts

We decided that it would be beneficial to list the different vocabularies found (e.g. RDF, RDFS, OWL, etc.) and each term (e.g. Description, Class, etc.) from the corresponding vocabularies, along with the reference counts for these terms. Note that this is a running count during the entire crawl. To accomplish this we created a new class called FILTERSINKVOCABULARY that implements the Sink class. We designed it to squeeze in front of the actual Sink, preprocessing each statement extracted by the Content Handler by accumulating the necessary statistics. It then passes on the unperturbed statements to the actual Sink for output. Once the crawl is compete, the method CLOSE() is called to output the statistics to the designated files.

The Content Handler (NxParser) is designed to communicate with the Sink through the Callback class, so the FILTERSINKVOCABULARY.NEWDATASET() method is called by each crawler lookup thread which creates a new Callback object called FILTERCALLBACK for each document to be parsed. Because the original NxParser resolves all Qualified Name (QName)

references to the full Universal Resource Identifier (URI) before sending to the Callback object via the processStatement(Node[] *nodes*) method, we needed to make changes to the RDFXMLParserBase class of the NxParser package. We modified the Callback class by adding a new method, processQName(String *qname*, Resource *uri*, URI *base)*, to allow the NxParser to pass each QName (e.g. rdf:Description, owl:sameAs, etc.) found to the FilterCallback object. It also passes the fully resolved URI for the QName, and the URI of the document being parsed.

We also created a new class, Vocabulary, to be used to record the vocabulary's full URI and a Map indexed by the terms with their associated accumulative counts. The FilterSinkVocabulary object retains the vocabulary statistics in a Map data structure indexed by the vocabulary's QName prefix (namespace) that references the corresponding vocabulary object.

4.2.2 Known and Unknown Vocabularies

Now that we can output vocabulary statistics, we decided to split them up into two categories (Maps): *knownVocabStats* and *unknownVocabStats*. We then created a set of known vocabularies to compare against. The following is our set of known vocabularies:

{bio, contact, dc, dcterms, doap, doc, dp, foaf, geo, org, owl, prv, rdf, rdfs, sioc, skos, vCard, xsd}

If the Map *knownVocabStats* contains the QName prefix, then the corresponding Vocabulary object is updated. If not, a new entry in *unknownVocabStats* was created with the QName prefix mapping to a newly instantiated Vocabulary object.

4.2.3 Document Statistics

In order to provide the same type of statistics on a per document basis, we needed to keep track of what vocabularies and terms were found for each document. This involved the creation of a new class, Document, to be used to record the document's full URI, the document's type (e.g. RDF, OWL, Other), and two Maps corresponding to the *knownVocabStats* and *unknownVocabStats* from above. Therefore, each document contains vocabulary

statistics that apply just to the document itself. These statistics are then used by the FilterSinkVocabulary class to be output to the document statistics file.

4.3 Write RDF to Triple Store

This capability exists in the original LDSpider program, so we wanted to test it. In order to test it, I downloaded and installed the OpenLink Software's Virtuoso Universal Server to my home computer. Once I configured and launched the Virtuoso Triple Server, I was able to connect LDSpider to the SPARQL/Update endpoint via the "`http:/localhost:8890`" URL. However, it was failing to load the RDF quads (subject-predicate-object-context), because of issues with the provenance data (document header information). This functionality is coded in the SinkSparul class, which implements the Sink class. We found that SinkSparul does not encapsulate blank nodes (URI prefix "_:") with "<" and ">" characters like it does the other nodes. Once we made changes to encapsulate blank nodes with "<" and ">" we were able to write RDF quads to the Virtuoso Triple Store and then query these quads using SPARQL queries.

4.4 Use Search Engine to Identify Seed URIs

Next, we decided it would be a good thing if we could use a search engine, specifically Google, to search for our seed URIs by file type (e.g. ".owl," ".rdf," etc.). This might lead us to documents in the Semantic Web universe that are disjointed from the Web of Linked Data cloud. This required adding to Main another command line option, "-g," with a required argument for the search configuration filename. This required adding a new method, findSeeds(File *searchList*), to read a search configuration file to get the file type and, optionally, topics used for the search.

After reading in the filetypes and/or search topics, a HTTP connection is made to the Google Search endpoint[4] with the query string "?v=1.0&***topic*** **filetype:*type***&userip=USERS-IP-ADDRESS" appended to the end of the URL. The Google Search endpoint then responds with the results in JavaScript Object Notation (JSON) format. This required downloading

[4] `https://ajax.googleapis.com/ajax/services/search/web`

open source code from the JSON in Java package[5] to integrate into LDSpider in order to correctly process the results. The results are read into a JSONOBJECT instance which is an unordered collection of name/value pairs. The JSONOBJECT data is then converted into a JSONARRAY which is then iterated through to find the values associated with the name "**url**". These values are added to the seed list.

4.5 Maximum Document Limit

Regardless of the type of crawl, the final results will always be the same if the termination is based on the depth of the crawl. Each level of depth is going to discover the same Frontier, just in a different order. If the crawl terminates between rounds, the document statistics and vocabulary statistics are going to be the same. It is the type of queue that determines the polling order; First-In-First-Out for breadth-first, ordered by size of pay-level domain queues, or ordered with OWL priority. Therefore, the crawl has to terminate mid-round, such that each type of crawl has a different queue to work off.

To do this, we implemented a maximum document limit such that the crawl will terminate when it reaches that limit, or the maximum depth limit, whichever comes first. This limit was added to MAIN as an extra argument (*maxdocs*) to each of the crawling type command line options. Changes then had to be made to the CRAWLER class and the appropriate LINKFILTER classes. The CRAWLER class has a method for each crawl type that loops around scheduling URIs for each round and starting the appropriate number of LOOKUPTHREAD objects. Code had to be added to terminate the crawl when the document limit has been reached. In addition, the appropriate LINKFILTER classes had to be updated to keep a running count of documents and to stop adding documents to the frontier when the limit has been reached.

4.6 Change Statistics from URI Based to Document Based

It soon became apparent that we would need to know the parent document for each child and a total of outgoing edges for each document parsed. This will help us view the results

[5] `http://www.json.org/java`

as an RDF graph. Unfortunately, LDSpider only deals with URIs and not documents. The Document class was only used by the FilterSinkVocabulary class for writing document statistics to the output file. In order to keep the original functionality of LDSpider intact, it was decided to only base crawls on documents when the vocabulary statistics command line option ("-v") was given.

Adding parent URI and outgoing link counter to the Document class is easy. However, this required wholesale changes to quite a few of the core components of LDSpider. A completely new Frontier class (DocumentFrontier) was developed to handle Documents instead of URIs. Also, this Frontier uses a FIFO queue instead of a set (i.e. BasicFrontier) or a map (i.e. RankedFrontier) data structure. Obviously, each type of SpiderQueue class had to be overhauled to handle queuing, adding and polling of Documents instead of URIs. This then led to an new LookupThread class, called DocumentLookupThread that polled the new SpiderQueue classes for Documents and retrieving the corresponding URI. That led to a new LinkFilterVocabulary) class that filters by Documents. Finally, a new crawler class (CrawlerDocs) was created to utilize these new and improved classes.

The LinkFilterVocabulary class was enhanced to not crawl to vocabulary namespaces. A set of vocabularies (initialized to known vocabularies) is updated every time a new vocabulary is found. This way, the crawler does not waste time and resources retrieving URIs for vocabulary terms.

Chapter 5

Giving Priority to OWL Documents

Previously, we talked about enhancing the LDSpider crawler to provide various statistics, to write RDF statements to a Triple Store, to query Google to find seed URIs, and to base the algorithms on documents instead of URIs. Now we will discuss the modifications to LDSpider that enable us to prove, or disprove, our stated hypothesis that by giving OWL documents priority during crawls we will locate more OWL documents than by other traversal methods. We believe it reasonable to postulate that OWL documents have proportionally more references to other kindred OWL documents than do non-OWL documents.

In this chapter, we will cover the two components of giving priority to OWL documents during a crawl. The first section describes the methods used to identify what is an OWL document. The second section explains how we implemented the OWL priority queue algorithm.

5.1 Identifying OWL Documents

The statement "give OWL documents priority" in our hypothesis implies that we know *before* we parse a document if it is an OWL document or not. Because it is impossible to know for sure what type of document is being referenced before we retrieve it, we needed to come up with some heuristics to *predict* if a document referenced by an URI will turn out to be an OWL document or not. These heuristics are described below:

Filename Extension: Obviously, the first indication that a URI points to an OWL file is by the filename extensions ".owl" or ".owx." Likewise, the first indication of a RDF file

is the extension ".rdf." Any other filename extensions are initially indicated as OTHER type. Note, however, that just because a document's filename contains the ".rdf" extension, it does not mean that the document's type is actually RDF. There are some instances where OWL documents have been given the ".rdf" or ".xml" extensions. Therefore, our first heuristic is to base a document's type upon instantiation according to its filename extension with the stipulation that the document's type can change anytime during parsing, unless the document is already flagged as an OWL document. Once a document has been identified as an OWL document, it will remain flagged as an OWL document.

OWL Language Constructs: The final determination of a document's type comes during parsing of the document. If the filename extension is ".owl" or ".owx" then the document is considered an OWL document regardless of parsing. If the document is considered an RDF or other type, then we look for certain OWL identifying constructs contained within. These constructs are considered to have "deeper" semantics with regard to ontologies, such that they are not borrowed by other vocabularies. For instance, the OWL:INTERSECTIONOF and OWL:UNIONOF constructs are often used in RDF and RDFS documents to construct new classes of resources, thus describing an ontology. On the other hand, the OWL:SAMEAS and OWL:EQUIVALENTCLASS constructs are often used in RDF and RDFS documents to denote equivalency between resources, which does not describe an ontology. Therefore, not all OWL language terms are used to identify OWL documents. The OWL language constructs that are used to determine that a document is of type OWL are listed below:

- owl:intersectionOf
- owl:unionOf
- owl:complementOf
- owl:allValuesFrom
- owl:someValuesFrom
- owl:hasValue
- owl:cardinality
- owl:minCardinality
- owl:maxCardinality
- owl:oneOf
- owl:DataRange
- owl: AsymmetricProperty

- owl:SymmetricProperty
- owl:IrreflexiveProperty
- owl:ReflexiveProperty
- owl:TransitiveProperty
- owl:disjointUnionOf
- owl:bottomObjectProperty
- owl:bottomDataProperty
- owl:topObjectProperty
- owl:topDataProperty
- owl:propertyChainAxiom
- owl:hasKey
- owl:maxQualifiedCardinality
- owl:minQualifiedCardinality
- owl:qualifiedCardinality
- owl:hasSelf
- owl:NegativePropertyAssertion
- owl:sourceIndividual
- owl:assertionProperty
- owl:targetIndividual
- owl:targetValue
- owl:onDataType
- owl:withRestrictions
- owl:datatypeComplementOf

5.2 Implementing an OWL Priority Queue

Now that we can identify potential OWL documents, it's time to give these documents priority during the crawl. To do this we added the following classes to the LDSpider project:

OwlFirstDocQueue: This class implements a Java priority queue (*queue*) of Documents, which orders the queue based on Document type. In order to instantiate this priority queue, we needed to provide a Java Comparator object which will compare two Documents and return a negative number if the first argument is less than the second, 0 if they are equal, and a positive number if the first argument is larger than the second.

This new Comparator class, called OwlPriorityComparator, implements a method compare(Document *arg0*, Document *arg1)* that takes two Document arguments and subtracts the document type ordinal value of the second argument from the document type ordinal value of the second argument. The document type is a Java enumeration defined as "public static enum *docTypes* {OWL, RDF, OTHER}." This means the type OWL has an ordinal value of 0, RDF has an ordinal value of 1, and OTHER has an ordinal value of 2. This Comparator allows the priority queue to put the OWL files

at the front, followed by RDF files, and then OTHER files. Therefore, chances are that when the crawl finishes, it has visited more OWL documents than any other because it has set aside the other documents for later.

OwlPriorityLookupThread: This class is modeled after the DocumentLookupThread class, however, it uses the Filename Extension heuristic from above to predict the document type from its corresponding URI. Therefore, we may have predicted what type of document it will be when it was added to the Frontier based on its parent document, but during retrieval we apply the heuristics all over again to the child document.

LinkFilterOwlPriority: This class is modeled after the LinkFilterVocabulary class, with one small change that makes a major impact. During document parsing, instead of adding document references to the Frontier, this class adds references identified in OWL documents back into the front of the current round's queue (OwlFirstDocQueue). For RDF and Other documents, the references are still added to the Frontier like normal. In this way, this crawling algorithm is comparable to a hybrid breadth-first/depth-first algorithm. It's breadth-first as long as we are processing anything other than OWL documents, but transitions to depth-first when OWL documents are encountered. Similar to strip mining for coal, the OWL priority queue strategy will scrape the surface looking for rich supplies, or veins, of OWL documents. Once a vein has been located, it drills down extracting all the OWL documents it can find. This should provide us more chances to find more OWL documents than by any other methods.

Chapter 6

Hypothesis Evaluation

Up to this point, we have discussed what it takes to develop and implement a crawler for the Semantic Web that attempts to locate OWL documents. We found an open source crawler that will traverse the web of Linked Data (LDSpider), but it does not focus on any one type of document. There is no doubt that it can find OWL documents during its crawl, especially if given the right seed documents. Because LDSpider employs only two types crawling strategies, breadth-first and load balanced, it will only appear to find OWL documents at random. We chose not to implement a depth-first strategy because that would be equivalent to the breadth-first strategy as far as finding OWL documents goes. However, if we focus on OWL documents themselves, and predict that they will lead us to more OWL documents, then we should be able to find more than by other crawling strategies.

Our final strategy is somewhat of a hybrid between breadth-first and depth-first strategies. It starts out as a breadth-first strategy going down the graph one level at a time. However, when it encounters an OWL document, it begins to follow its edges down the graph as long as it keeps finding more OWL documents. Once it has reached a leaf node in terms of an OWL only subgraph, it then returns to the breadth-first strategy at the original level that it left off at. Hence the hybrid reference to describe this strategy.

Note that the evaluation can be skewed one way or the other based on the starting point and the crawl size. For example, if you start in a high RDF density part of the Linked Data Cloud for a relatively small crawl, you will find roughly the same amount of OWL documents regardless of crawl method. If you go the other way, you could start in a highly coupled OWL part of the Linked Data Cloud for the same size crawl, where you will probably

find a larger number of OWL documents regardless of crawl method. In either case, the numbers can be attributed to the starting point, but the variance between the crawls should be limited. Of course, the bigger the crawl the more the variance. For the test runs used in this evaluation, we used Google search to find a somewhat random and diverse set of seed documents that should not lean toward any particular crawling strategy. See Appendix B for the list of seed URIs used for this evaluation.

Now it is time to evaluate our "hybrid" to see if our hypothesis is true. Does implementing an OWL priority queue locate more OWL documents than do other strategies? The following sections will delve into this evaluation by first looking at the number of OWL vocabulary terms found. If we find more OWL vocabulary terms using this priority queue, then it follows that we found more OWL documents. Finally, we determine whether we found significantly more OWL documents using our OWL priority queue strategy than by breadth-first and load-balanced strategies.

6.1 OWL Vocabulary Statistics

Let's begin this evaluation by taking a look at the vocabulary statistics that we've been gathering during our test crawls. At the end of each run, the LDSpider program writes the accumulated statistics to an output file. This output file begins by reporting the total number of vocabularies and the total number of vocabulary terms found during the crawl. The output file then lists both *known* vocabularies (vocabularies we expected to find), and *unknown* vocabularies (vocabularies we did not expect). For each vocabulary, it shows the total number of terms found, then lists each term with its associated counts. After it lists all the terms, it provides a subtotal of all the terms found for each vocabulary. See Appendix C for a sample of the vocabulary statistics output from a small 20 document crawl employing the OWL priority queue strategy.

These statistics allow us to identify all the different vocabularies that are found, and how often their vocabulary terms are used. For the purpose of our thesis, let's take a look at the OWL vocabulary terms found during our test runs of varying lengths (100, 200, 500, 1,000, 10,000, 20,000 and 50,000 documents) using all three crawling strategies. The results are tabulated in Table 6.1 below:

Crawl Size	Breadth-First	Load-Balanced	OWL-Priority-Queue
100	173,800	170,434	171,716
200	174,645	175,720	172,520
500	175,303	217,664	174,244
1,000	176,412	254,615	177,385
10,000	199,790	239,819	200,411
20,000	233,098	363,387	205,561
50,000	1,046,263	1,211,357	1,054,126
Total	2,179,311	2,632,996	2,155,963

Table 6.1: Number of OWL Vocabulary Terms Parsed During Crawl

As you can see, the number of OWL terms for each strategy did not vary much for the smaller crawls (100 and 200 documents), but seems to vary widely once we reached the 500 document crawl, especially for the load-balanced strategy. The lack of variance from the smaller crawls is likely due to the fact that for the first few rounds, the list of documents crawled are likely to be very similar regardless of strategy. The big surprise is that the leader, by a wide margin, in finding OWL terms for the larger crawls is the load-balanced strategy, while the breadth-first and OWL priority strategies seem to have very little variance. These vocabulary statistics may be offering a hint of what is about to come.

6.2 OWL Document Statistics

Now, the pièce-de-résistance; the determination of whether we found a significant amount more OWL documents using our OWL priority queue strategy than by breadth-first and load-balanced strategies. At the end of each run, the LDSpider program writes the accumulated document statistics to an output file. This output file begins by reporting the total number of documents crawled, followed by the total number of documents found by document type (i.e. OWL, RDF, or Other). This is then followed by the statistics for each document found. This is comprised of the document's URI, the number of outgoing edges, and the URI of the parent node in the graph. This information is immediately followed by

the vocabulary statistics for each document in the same format as above for the vocabulary statistics output file. See Appendix D for a sample of the document statistics output from a small 20 document crawl employing the OWL priority queue strategy.

These document statistics provide the necessary information to determine how many OWL documents are located during a crawl, in addition to statistics to glean information necessary to visualize the Linked Data graph. Please note, however, that the outgoing edges is a count of how many outgoing URIs were found, not necessarily crawled. Not all these outgoing links were actually retrieved because of the Link Filter or Fetch Filters (i.e. JPG and HTML files are not retrieved). To determine the significance of our results, let's take a look at the hit rate (percentage of OWL documents located) during our test runs of varying lengths (100, 200, 500, 1,000, 10,000, 20,000 and 50,000 documents) using all three crawling strategies. The results are tabulated in Table 6.2 below:

Crawl Size	Breadth-First	Load-Balanced	OWL-Priority-Queue
100	26.00	28.00	22.00
200	14.00	16.50	12.00
500	5.80	8.60	5.00
1,000	2.90	4.90	2.50
10,000	0.31	0.44	1.03
20,000	0.18	0.25	0.52
50,000	0.26	0.75	0.47
Mean	7.06	8.49	6.22
Large Crawl	0.25	0.48	0.67

Table 6.2: Percentage of OWL Documents Located During Crawl

As you can see from the table, the hit rate is highest for the smaller crawls as the seed OWL documents comprise a large percentage of the total. As the size of the crawls increase the hit rate drops to less than 1%. This is due to the nature of OWL documents, which are used to model domains of knowledge, whereas RDF and RDFS is used to describe instances of data pertaining to a given domain. Therefore, it follows that there will be much more instances of RDF and RDFS than OWL documents.

Unfortunately, these results indicate that the OWL priority queue algorithm does *not* necessarily find more OWL documents than either the breadth-first or load-balanced algorithms. In fact, the mean of the percentage of OWL documents found for each algorithm shows that the OWL priority algorithm found the least. From the data shown for the seven different sized crawls, with no variation in the seed list, the load-balanced algorithm fared the best with a mean of 8.49%, followed by breadth-first with a mean of 7.06%, with OWL priority bringing up the rear with 6.22%.

This was definitely not the results we were expecting, which begs the question "Why did the OWL priority algorithm do so poorly?" During analysis of the smaller crawls (100 to 1000 documents), we found that the OWL priority algorithm would complete before it processed all the documents in the seed list, which caused it to miss some of the OWL documents in the seed list. This was because it would begin digging down from the first few OWL documents in the seed list. It appears that once the crawler began the depth-first method, the links from the OWL documents did not reference very many other OWL documents. Thus the crawl ended before it could parse all the seed documents, thus missing out on potentially many more OWL documents. Whereas the other algorithms always processed every document in the seed list.

For the larger crawls (10,000 to 50,000 documents), it appears that the OWL priority algorithm does much better. The one aberration is the 50,000 document load-balanced crawl which found 376 OWL documents versus 235 for the OWL priority crawl. Analysis of that crawl showed that a very high percentage of OWL documents found were referenced by other OWL documents which follows the logic used in our hypothesis. That begs the question, "So, why didn't the OWL priority crawl find the same OWL documents?" Further analysis showed that many of the parent OWL documents for the load-balanced crawl did not not seem to reference the same child document in the OWL priority crawl. It appears that at the time of the OWL priority crawl the child documents did not parse correctly, possibly because the host server was down. During much of the testing, we found that often many sites were unreachable and/or the connection timed out. In fact, it was difficult to duplicate results from one test to the other, especially when some of the larger crawls would take as many as four to six *days* to complete. Because of this analysis, the assumption is made

that if all the servers were available during the OWL priority 50,000 document crawl as was during the load-balanced crawl, then we would have seen much better results.

Therefore, for larger than 1000 document crawls, these results do seem to show that the OWL priority queue algorithm does locate more OWL documents than either the breadth-first or load-balanced algorithms. The key question is, "Are these results significant enough to provide validation to our hypothesis?" With the sample size given for the larger crawls (10,000, 20,000 and 50,000 documents), and no variation in the seed list, we may have results that can be expected from a statistically random sample. In order to have confidence that our results are significant enough to give us confidence in the validity of our hypothesis, we believe that the hit rate using the OWL priority queue algorithm should be at least one standard deviation greater than the mean of the hit rate by the other algorithms.

Without going into the details of the statistical calculations, we show that the mean of the percentages, or hit rate, shown in Table 6.2 for all of the larger crawls using breadth-first and load-balanced algorithms is 0.37%, with a standard deviation of 0.19%. Therefore, the band of the first standard deviation goes from 0.18% to 0.56%. Because the mean of the percentages for the larger OWL priority crawls is 0.67%, which is greater than 1 standard deviation from the mean of the breadth-first and load-balance crawls, this appears to be a significant enough difference that we can consider it a validation of our hypothesis. Unfortunately, because of time constraints for the larger crawls, we were unable to provide a larger sample by increasing the size of the crawls or by repeating these test crawls. Having said that, we ran many test crawls while tweaking the software, and all indications from those test runs were consistent with these set of data provided here. Again, in almost all the test cases, the crawls larger than 1000 documents indicated that the OWL priority strategy retrieved the most OWL documents.

In summary, it appears that from our research that by developing a Semantic Web crawler that gives OWL documents priority during crawling does, in fact, find more OWL documents than other types of crawls, as long as the crawl large enough. Note that this is not an absolute truth. There are many conditions (i.e. seed list, server availability, etc.) that can affect the outcome, but as a general rule our hypothesis holds true.

Chapter 7

Future Research

As with all research, there is always more research than can be derived from it. In this case, we did not have the time to accomplish really large crawls (greater than 50,000 documents) that might lead to more interesting results. Larger crawls might lead us to sets of OWL documents that were previously undiscovered because they are outside the web of Linked Data. Also, it might fruitful to investigate the possibility of better heuristics to identify OWL documents. If a better way of predicting OWL documents is discovered, then it may be more efficient to skip the retrieval of other types of documents and stick to just OWL documents, thus increasing the hit rate.

In addition, it seems that this crawler, as written, has performance issues. One thing to consider would be to have two queues for each algorithm. One to use as the frontier and the other as the document queue. These queues would swap functions after each round; The frontier would become the queue, and the empty queue would then become the frontier. This would eliminate the frontier class, thus eliminating the moving of documents from the frontier to the queue at the end of each round. Another thing to consider would be to write the vocabulary and document statistics out to a file periodically to avoid the massive overhead of very large Java map and queue data structures.

Also, we were hoping to delve into providing reasoning support for the RDF quads using the Virtuoso Triple Store. This would allow us to automatically infer more information from the RDF statements that we found during the crawl. This would increase our knowledge base of the ontologies that we found. Perhaps someone could take on this research task in the future.

Chapter 8

Conclusion

So there you have it, this thesis was a narrative describing our research into, and development of, a Semantic Web Crawler that will locate OWL documents. We believed that by developing a web crawler that implements a crawling algorithm using a queue that gives *predicted* OWL documents priority, that we would find more OWL files than by any other crawling algorithms. The rationale behind this hypothesis is that we believe OWL documents have proportionally more links to OWL documents than to other types of documents. This thesis set out to provide validation to our hypothesis.

We began with a short introduction, followed by a chapter on the background information necessary to understand and appreciate this research. We discussed the software technologies used in this research like the Semantic Web, XML, RDF, RDFS, OWL and the web of Linked Data. We also discussed the Virtuoso Triple Store and Eclipse software used for development and testing of our web crawler.

We then detailed our search for an appropriate open source web crawler to provide the base software needed for this research. This included the reasoning used to pick LDSpider as our crawler of choice.

Next came a couple of chapters discussing the enhancements and modifications we made to the LDSpider program in order to reach our conclusions. We provided an in-depth look into how, and why, we implemented the OWL priority queue algorithm. We showed what Java classes we created, and the original ones that we had to modify.

The next chapter was our evaluation. We showed results from our testing, and calculated the mean and standard deviation for the larger breadth-first and load-balanced crawl hit

rates. The mean of the hit rates for the larger crawls using the OWL priority strategy was then compared against the previous mean and standard deviation, showing that there was a significant enough difference to give validity to our hypothesis. Further research should find similar results.

In retrospect, the true goal of this thesis was not necessarily to break new ground in the development of web crawlers or the Semantic Web, but to gain experience and learn from doing research in this area. I am able to take away from this thesis the knowledge of the Semantic Web and its components (i.e. XML, RDF, RDFS, OWL, SPARQL, etc.) and its use in the real world. During the testing of this crawler, I had to learn how to "read" RDF/XML and interpret which URIs needed to be retrieved and which ones didn't. The original LDSpider program would follow all URIs regardless of whether they referenced a vocabulary term or not. This research also gave me insight into the world of web crawlers and web crawling techniques. In addition, I greatly enhanced my Java programming skills during this research. I'm sure I more than doubled my experience of programming in Java.

In summary, the learning curve for this research was steep, but the experience is invaluable, and cannot be duplicated in a classroom. I will always take with me the knowledge and understanding that I gained during this effort. I have great hope and confidence that I will call upon this experience and knowledge in my future endeavors.

Appendices

Appendix A

LDSpider Directory Structure

```
source/com:
drwxr-xr-x  4 ronkoron     136B Jun  3 01:03 ontologycentral

source/com/ontologycentral:
drwxr-xr-x  13 ronkoron     442B May 16 21:57 ldspider

source/com/ontologycentral/ldspider:
-rw-r--r--@  1 ronkoron      11K May 16 23:00 Crawler.java
-rw-r--r--@  1 ronkoron     2.1K Aug 25 12:26 CrawlerConstants.java
-rw-r--r--   1 ronkoron      13K Aug 12 00:38 CrawlerDocs.java
-rw-r--r--@  1 ronkoron     1.4K Jun 17  2010 LDSpider_LogUtil.java
-rw-r--r--@  1 ronkoron      21K Jun 17 18:08 Main.java
-rw-r--r--   1 ronkoron     7.6K May 19 18:48 Statistician.java
drwxr-xr-x   7 ronkoron     238B May 13 19:09 frontier
drwxr-xr-x   7 ronkoron     238B Oct 18  2011 hooks
drwxr-xr-x   9 ronkoron     306B May 15 23:37 http
drwxr-xr-x  11 ronkoron     374B May 15 23:04 queue
drwxr-xr-x   5 ronkoron     170B Nov  6  2011 tld

source/com/ontologycentral/ldspider/frontier:
-rw-r--r--@ 1 ronkoron     707B Mar 19  2010 BasicFrontier.java
-rw-r--r--@ 1 ronkoron     1.9K Jun  6  2010 Frontier.java
-rw-r--r--  1 ronkoron     1.8K May 19 17:19 OwlRankedFrontier.java
-rw-r--r--@ 1 ronkoron     1.2K Mar 19  2010 RankedFrontier.java
drwxr-xr-x  5 ronkoron     170B May 13 19:09 documentfrontier

source/com/ontologycentral/ldspider/frontier/documentfrontier:
-rw-r--r--  1 ronkoron     2.2K Aug 12 14:50 DocumentFrontier.java
-rw-r--r--  1 ronkoron     1.2K May 31 17:42 DocumentRankedFrontier.java
-rw-r--r--  1 ronkoron     2.1K May 31 17:42 OwlDocumentFrontier.java
```

```
source/com/ontologycentral/ldspider/hooks:
drwxr-xr-x   7 ronkoron     238B Nov 30  2011 content
drwxr-xr-x   6 ronkoron     204B Oct 18  2011 error
drwxr-xr-x   7 ronkoron     238B Oct 18  2011 fetch
drwxr-xr-x  10 ronkoron     340B May  1 23:50 links
drwxr-xr-x   8 ronkoron     272B Oct 18  2011 sink

source/com/ontologycentral/ldspider/hooks/content:
-rw-r--r--@ 1 ronkoron     1.0K Jun  2  2010 ContentHandler.java
-rw-r--r--@ 1 ronkoron     5.1K Jun  2  2010 ContentHandlerAny23.java
-rw-r--r--@ 1 ronkoron     1.1K Jun  2  2010 ContentHandlerNx.java
-rw-r--r--@ 1 ronkoron     1.6K May 26 19:04 ContentHandlerRdfXml.java
-rw-r--r--@ 1 ronkoron     999B Jun  2  2010 ContentHandlers.java

source/com/ontologycentral/ldspider/hooks/error:
-rw-r--r--@ 1 ronkoron     574B Jun  6  2010 ErrorHandler.java
-rw-r--r--@ 1 ronkoron     628B Jun  6  2010 ErrorHandlerDummy.java
-rw-r--r--@ 1 ronkoron     6.5K Jan 23  2012 ErrorHandlerLogger.java
-rw-r--r--@ 1 ronkoron     285B Jan 20  2010 ObjectThrowable.java

source/com/ontologycentral/ldspider/hooks/fetch:
-rw-r--r--@ 1 ronkoron     202B Oct 16  2009 FetchFilter.java
-rw-r--r--@ 1 ronkoron     245B Jan 19  2010 FetchFilterAllow.java
-rw-r--r--@ 1 ronkoron     245B Jan 19  2010 FetchFilterDeny.java
-rw-r--r--@ 1 ronkoron     1.1K May 20 00:17 FetchFilterRdfXml.java
-rw-r--r--@ 1 ronkoron     454B Sep  3  2010 FetchFilterSuffix.java

source/com/ontologycentral/ldspider/hooks/links:
-rw-r--r--@ 1 ronkoron     596B May  7 23:49 LinkFilter.java
-rw-r--r--@ 1 ronkoron     3.5K May 10 22:16 LinkFilterDefault.java
-rw-r--r--@ 1 ronkoron     1.7K May  7 23:49 LinkFilterDomain.java
-rw-r--r--@ 1 ronkoron     1.3K May 10 22:12 LinkFilterDummy.java
-rw-r--r--  1 ronkoron     9.9K Aug 22 00:21 LinkFilterOwlPriority.java
-rw-r--r--@ 1 ronkoron     1.6K May  7 23:49 LinkFilterPrefix.java
-rw-r--r--@ 1 ronkoron     1.7K May  7 23:49 LinkFilterSelect.java
-rw-r--r--  1 ronkoron     6.9K Aug 22 00:21 LinkFilterVocabulary.java

source/com/ontologycentral/ldspider/hooks/sink:
-rw-r--r--@ 1 ronkoron     1.1K Jun  2  2010 Provenance.java
-rw-r--r--@ 1 ronkoron     441B Nov  5  2011 Sink.java
-rw-r--r--@ 1 ronkoron     1.2K Apr  8 01:58 SinkCallback.java
-rw-r--r--@ 1 ronkoron     794B Apr 18 00:31 SinkDummy.java
-rw-r--r--@ 1 ronkoron     7.0K Apr 18 00:37 SinkSparul.java
drwxr-xr-x  4 ronkoron     136B May 10 22:17 filter
```

```
source/com/ontologycentral/ldspider/hooks/sink/filter:
-rw-r--r--@ 1 ronkoron     2.2K May 10 22:55 FilterSinkPredicate.java
-rw-r--r--  1 ronkoron      12K May 28 15:39 FilterSinkVocabulary.java

source/com/ontologycentral/ldspider/http:
-rw-r--r--@ 1 ronkoron     4.0K Sep  3  2010 ConnectionManager.java
-rw-r--r--  1 ronkoron     6.2K Aug 19 09:58 DocumentLookupThread.java
-rw-r--r--@ 1 ronkoron     2.8K Nov 21  2011 Headers.java
-rw-r--r--@ 1 ronkoron     5.9K May 11 00:17 LookupThread.java
-rw-r--r--  1 ronkoron     6.3K May 31 12:21 OwlPriorityLookupThread.java
drwxr-xr-x  6 ronkoron     204B Oct 18  2011 internal
drwxr-xr-x  4 ronkoron     136B Oct 18  2011 robot

source/com/ontologycentral/ldspider/http/internal:
-rw-r--r--@ 1 ronkoron     1.2K Jun 23  2010 CloseIdleConnectionThread.java
-rw-r--r--@ 1 ronkoron     2.2K Jan  5  2010 GzipDecompressingEntity.java
-rw-r--r--@ 1 ronkoron     1.4K Jan  5  2010 HttpRequestRetryHandler.java
-rw-r--r--@ 1 ronkoron     2.8K Jan  5  2010 ResponseGzipUncompress.java

source/com/ontologycentral/ldspider/http/robot:
-rw-r--r--@ 1 ronkoron     2.7K Aug 22 21:34 Robot.java
-rw-r--r--@ 1 ronkoron     1.3K Jan 20  2010 Robots.java

source/com/ontologycentral/ldspider/queue:
-rw-r--r--  1 ronkoron     3.6K Aug 12 01:36 BreadthFirstDocQueue.java
-rw-r--r--@ 1 ronkoron     5.8K May  7 23:08 BreadthFirstQueue.java
-rw-r--r--  1 ronkoron     6.5K Aug 26 13:14 LoadBalancingDocQueue.java
-rw-r--r--@ 1 ronkoron     5.6K Jul  4 17:19 LoadBalancingQueue.java
-rw-r--r--  1 ronkoron     4.2K Aug 22 22:38 OwlFirstDocQueue.java
-rw-r--r--  1 ronkoron     5.9K May 19 17:17 OwlFirstQueue.java
-rw-r--r--@ 1 ronkoron     827B Aug 12 12:26 Redirects.java
-rw-r--r--@ 1 ronkoron     2.5K May 19 18:39 SpiderQueue.java
-rw-r--r--  1 ronkoron     2.7K Aug  1 23:54 SpiderQueueDoc.java

source/com/ontologycentral/ldspider/tld:
-rw-r--r--@ 1 ronkoron     2.6K Jun 17  2010 Tld.java
-rw-r--r--@ 1 ronkoron     7.7K Jan 22  2012 TldManager.java
-rw-r--r--@ 1 ronkoron      36K Dec  1  2009 tld.dat.txt
drwxr-xr-x  4 ronkoron     136B Jun  3 01:03 com

drwxr-xr-x  4 ronkoron     136B Mar  5 23:09 org

mancp/org:
drwxr-xr-x@ 18 ronkoron     612B Mar  6 23:12 json
drwxr-xr-x   3 ronkoron     102B Jan 16  2012 osjava
```

```
mancp/org/json:
-rwxr-xr-x@ 1 ronkoron      10K Mar  6 23:12 CDL.java
-rwxr-xr-x@ 1 ronkoron     6.5K Feb 16  2012 Cookie.java
-rwxr-xr-x@ 1 ronkoron     3.3K Feb 16  2012 CookieList.java
-rwxr-xr-x@ 1 ronkoron     5.8K Feb 16  2012 HTTP.java
-rwxr-xr-x@ 1 ronkoron     2.4K Feb 16  2012 HTTPTokener.java
-rw-r--r--@ 1 ronkoron      29K Apr  8 23:19 JSONArray.java
-rwxr-xr-x@ 1 ronkoron     706B Feb 16  2012 JSONException.java
-rwxr-xr-x@ 1 ronkoron      17K Feb 16  2012 JSONML.java
-rwxr-xr-x@ 1 ronkoron      54K Feb 16  2012 JSONObject.java
-rwxr-xr-x@ 1 ronkoron     733B Feb 16  2012 JSONString.java
-rwxr-xr-x@ 1 ronkoron     3.2K Feb 16  2012 JSONStringer.java
-rw-r--r--@ 1 ronkoron      13K Feb 16  2012 JSONTokener.java
-rwxr-xr-x@ 1 ronkoron      10K Feb 16  2012 JSONWriter.java
-rwxr-xr-x  1 ronkoron     2.3K Feb 16  2012 README
-rwxr-xr-x@ 1 ronkoron      17K Feb 16  2012 XML.java
-rwxr-xr-x@ 1 ronkoron      11K Feb 16  2012 XMLTokener.java

mancp/org/osjava:
drwxr-xr-x  10 ronkoron     340B Jan 16  2012 norbert

mancp/org/osjava/norbert:
-rw-r--r--  1 ronkoron     2.1K Jan 16  2012 AbstractRule.java
-rw-r--r--  1 ronkoron     2.1K Jan 16  2012 AllowedRule.java
-rw-r--r--  1 ronkoron     2.1K Jan 16  2012 DisallowedRule.java
-rw-r--r--  1 ronkoron     9.3K May 11 00:28 NoRobotClient.java
-rw-r--r--  1 ronkoron     2.0K Jan 16  2012 NoRobotException.java
-rw-r--r--  1 ronkoron     1.9K Jan 16  2012 Rule.java
-rw-r--r--  1 ronkoron     2.9K Jan 16  2012 RulesEngine.java

mancp/org/osjava/norbert/.svn/prop-base:

mancp/org/osjava/norbert/.svn/props:

mancp/org/osjava/norbert/.svn/text-base:
-r--r--r--  1 ronkoron     2.1K Jan 16  2012 AbstractRule.java.svn-base
-r--r--r--  1 ronkoron     2.1K Jan 16  2012 AllowedRule.java.svn-base
-r--r--r--  1 ronkoron     2.1K Jan 16  2012 DisallowedRule.java.svn-base
-r--r--r--  1 ronkoron     9.1K Jan 16  2012 NoRobotClient.java.svn-base
-r--r--r--  1 ronkoron     2.0K Jan 16  2012 NoRobotException.java.svn-base
-r--r--r--  1 ronkoron     1.9K Jan 16  2012 Rule.java.svn-base
-r--r--r--  1 ronkoron     2.9K Jan 16  2012 RulesEngine.java.svn-base

NxParser/lib:
-rw-r--r--  1 ronkoron      35K Aug 15  2011 commons-cli-1.1.jar
```

```
-rw-r--r--  1 ronkoron     281K Aug 15  2011 htmlparser.jar
-rw-r--r--  1 ronkoron      74B Dec 22  2011 maven-info.txt

NxParser/org:
drwxr-xr-x  4 ronkoron     136B Jan 21  2012 semanticweb

NxParser/org/semanticweb:
drwxr-xr-x  6 ronkoron     204B Jan 21  2012 yars
drwxr-xr-x  3 ronkoron     102B Jan 21  2012 yars2

NxParser/org/semanticweb/yars:
drwxr-xr-x  46 ronkoron     1.5K May 14 21:52 nx
drwxr-xr-x  36 ronkoron     1.2K May 13 14:38 stats
drwxr-xr-x   9 ronkoron     306B May 13 14:38 tld
drwxr-xr-x  48 ronkoron     1.6K May 13 14:38 util

NxParser/org/semanticweb/yars/nx:
-rw-r--r--   1 ronkoron     4.6K Dec 22  2011 BNode.java
-rw-r--r--   1 ronkoron     2.0K Dec 22  2011 BooleanLiteral.java
-rw-r--r--   1 ronkoron     4.5K Dec 22  2011 DateLiteral.java
-rw-r--r--   1 ronkoron     4.5K Dec 22  2011 DateTimeLiteral.java
-rw-r--r--   1 ronkoron     3.1K May 20 00:06 Document.java
-rw-r--r--   1 ronkoron     8.3K Jan  3  2012 Literal.java
-rw-r--r--   1 ronkoron     643B Dec 22  2011 Node.java
-rw-r--r--   1 ronkoron     7.5K Dec 22  2011 NodeComparator.java
-rw-r--r--   1 ronkoron     2.3K Dec 22  2011 Nodes.java
-rw-r--r--   1 ronkoron     5.4K Dec 22  2011 NumericLiteral.java
-rw-r--r--   1 ronkoron     2.9K Dec 22  2011 Quad.java
-rw-r--r--   1 ronkoron     5.8K May 19 16:44 Resource.java
-rw-r--r--   1 ronkoron     3.0K Dec 22  2011 Triple.java
-rw-r--r--   1 ronkoron     805B Dec 22  2011 Unbound.java
-rw-r--r--   1 ronkoron     2.0K Dec 22  2011 Variable.java
-rw-r--r--   1 ronkoron     1.8K May 19 22:56 Vocabulary.java
drwxr-xr-x   6 ronkoron     204B May 13 14:38 clean
drwxr-xr-x  51 ronkoron     1.7K May 13 14:38 cli
drwxr-xr-x  22 ronkoron     748B May 13 14:38 dt
drwxr-xr-x  12 ronkoron     408B May 13 14:38 file
drwxr-xr-x  13 ronkoron     442B May 13 14:38 filter
drwxr-xr-x   4 ronkoron     136B May 13 14:38 hash
drwxr-xr-x   6 ronkoron     204B May 13 14:38 mem
drwxr-xr-x  40 ronkoron     1.3K May 13 14:38 namespace
drwxr-xr-x  14 ronkoron     476B May 13 14:38 parser
drwxr-xr-x   4 ronkoron     136B May 13 14:38 reorder
drwxr-xr-x   9 ronkoron     306B May 13 14:38 sort
drwxr-xr-x   4 ronkoron     136B May 13 14:38 util
```

```
NxParser/org/semanticweb/yars/nx/clean:
-rw-r--r--  1 ronkoron     8.7K Dec 22  2011 Cleaner.java
-rw-r--r--  1 ronkoron     636B Dec 22  2011 HTMLTextExtractor.java

NxParser/org/semanticweb/yars/nx/cli:
-rw-r--r--  1 ronkoron     2.3K Dec 22  2011 CheckSorted.java
-rw-r--r--  1 ronkoron     2.6K Dec 22  2011 Clean.java
-rw-r--r--  1 ronkoron     3.0K Dec 22  2011 CleanXML.java
-rw-r--r--  1 ronkoron     4.4K Dec 22  2011 CreateFiles.java
-rw-r--r--  1 ronkoron     2.4K Dec 22  2011 CreateNtriples.java
-rw-r--r--  1 ronkoron     2.6K Dec 22  2011 CreateQuad.java
-rw-r--r--  1 ronkoron     5.2K Dec 22  2011 CreateRDFXML.java
-rw-r--r--  1 ronkoron     3.0K Dec 22  2011 CreateSQL.java
-rw-r--r--  1 ronkoron     2.6K Dec 22  2011 FixBNodes.java
-rw-r--r--  1 ronkoron     2.3K Dec 22  2011 GetNSs.java
-rw-r--r--  1 ronkoron     3.7K Dec 22  2011 GetPlds.java
-rw-r--r--  1 ronkoron     4.0K Dec 22  2011 GetTBox.java
-rw-r--r--  1 ronkoron     3.0K Dec 22  2011 GetURIs.java
-rw-r--r--  1 ronkoron     3.1K Dec 22  2011 Head.java
-rw-r--r--  1 ronkoron     8.5K Dec 22  2011 Main.java
-rw-r--r--  1 ronkoron     4.9K Dec 22  2011 MergeSort.java
-rw-r--r--  1 ronkoron     2.9K Dec 22  2011 Parse.java
-rw-r--r--  1 ronkoron     2.5K May 10 23:55 ParseRDFXML.java
-rw-r--r--  1 ronkoron     2.8K Dec 22  2011 PickLabels.java
-rw-r--r--  1 ronkoron     2.9K Dec 22  2011 Reorder.java
-rw-r--r--  1 ronkoron     2.5K Dec 22  2011 Sample.java
-rw-r--r--  1 ronkoron     5.4K Dec 22  2011 Sort.java
-rw-r--r--  1 ronkoron     4.7K Dec 22  2011 Split.java
drwxr-xr-x  5 ronkoron     170B May 13 14:38 factory
-rw-r--r--  1 ronkoron     2.3K Dec 22  2011 voiD.java

NxParser/org/semanticweb/yars/nx/cli/factory:
-rw-r--r--  1 ronkoron      11K Dec 22  2011 DOMConfigFileHandler.java

NxParser/org/semanticweb/yars/nx/dt:
-rw-r--r--   1 ronkoron     1.7K Dec 22  2011 Datatype.java
-rw-r--r--   1 ronkoron     6.1K Jan  3  2012 DatatypeFactory.java
-rw-r--r--   1 ronkoron     1.1K Dec 22  2011 DatatypeParseException.java
-rw-r--r--   1 ronkoron     423B Dec 22  2011 UnsupportedDatatypeException.
                                              java
-rw-r--r--   1 ronkoron     5.6K Dec 22  2011 XMLRegex.java
-rw-r--r--   1 ronkoron      10K Apr 29 16:04 XSDDatatypeMap.java
drwxr-xr-x   6 ronkoron     204B May 13 14:38 binary
drwxr-xr-x   4 ronkoron     136B May 13 14:38 bool
drwxr-xr-x  22 ronkoron     748B May 13 14:38 datetime
```

```
drwxr-xr-x  34 ronkoron     1.1K May 13 14:38 numeric
drwxr-xr-x  16 ronkoron     544B May 13 14:38 string
drwxr-xr-x   4 ronkoron     136B May 13 14:38 uri
drwxr-xr-x   5 ronkoron     170B May 13 14:38 xml

NxParser/org/semanticweb/yars/nx/dt/binary:
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDBase64Binary.java
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 XSDHexBinary.java

NxParser/org/semanticweb/yars/nx/dt/bool:
-rw-r--r--  1 ronkoron     980B Dec 22  2011 XSDBoolean.java

NxParser/org/semanticweb/yars/nx/dt/datetime:
-rw-r--r--  1 ronkoron      22K Dec 22  2011 ISO8601Parser.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDDate.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 XSDDateTime.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 XSDDateTimeStamp.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDGDay.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDGMonth.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDGMonthDay.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDGYear.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDGYearMonth.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 XSDTime.java

NxParser/org/semanticweb/yars/nx/dt/numeric:
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 XSDByte.java
-rw-r--r--  1 ronkoron     1.8K Dec 22  2011 XSDDecimal.java
-rw-r--r--  1 ronkoron     2.2K Dec 22  2011 XSDDouble.java
-rw-r--r--  1 ronkoron     2.2K Dec 22  2011 XSDFloat.java
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 XSDInt.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 XSDInteger.java
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 XSDLong.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 XSDNegativeInteger.java
-rw-r--r--  1 ronkoron     1.5K Dec 22  2011 XSDNonNegativeInteger.java
-rw-r--r--  1 ronkoron     1.5K Dec 22  2011 XSDNonPositiveInteger.java
-rw-r--r--  1 ronkoron     1.6K Dec 22  2011 XSDPositiveInteger.java
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 XSDShort.java
-rw-r--r--  1 ronkoron     1.8K Dec 22  2011 XSDUnsignedByte.java
-rw-r--r--  1 ronkoron     1.8K Dec 22  2011 XSDUnsignedInt.java
-rw-r--r--  1 ronkoron     1.9K Dec 22  2011 XSDUnsignedLong.java
-rw-r--r--  1 ronkoron     1.8K Dec 22  2011 XSDUnsignedShort.java

NxParser/org/semanticweb/yars/nx/dt/string:
-rw-r--r--  1 ronkoron     1.8K Dec 22  2011 XSDLanguage.java
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 XSDNCName.java
```

```
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 XSDNMToken.java
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 XSDName.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDNormalisedString.java
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 XSDString.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 XSDToken.java

NxParser/org/semanticweb/yars/nx/dt/uri:
-rw-r--r--  1 ronkoron     1.0K Dec 22  2011 XSDAnyURI.java

NxParser/org/semanticweb/yars/nx/dt/xml:
-rw-r--r--  1 ronkoron     3.3K Dec 22  2011 RDFXMLLiteral.java

NxParser/org/semanticweb/yars/nx/file:
-rw-r--r--  1 ronkoron     1.8K Dec 22  2011 FileInput.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 NxGzInput.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 NxInput.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 RDFXMLGzInput.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 RDFXMLInput.java

NxParser/org/semanticweb/yars/nx/filter:
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 FilterIterator.java
-rw-r--r--  1 ronkoron     4.1K Dec 22  2011 NodeFilter.java

NxParser/org/semanticweb/yars/nx/hash:
-rw-r--r--  1 ronkoron     4.6K Dec 22  2011 HashLibrary.java

NxParser/org/semanticweb/yars/nx/mem:
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 LowMemorySniffer.java
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 MemoryManager.java

NxParser/org/semanticweb/yars/nx/namespace:
-rw-r--r--  1 ronkoron     581B Dec 22  2011 DC.java
-rw-r--r--  1 ronkoron     418B Dec 22  2011 DCTERMS.java
-rw-r--r--  1 ronkoron     4.9K Dec 22  2011 FOAF.java
-rw-r--r--  1 ronkoron     512B Dec 22  2011 GEO.java
-rw-r--r--  1 ronkoron     4.6K Dec 22  2011 HTTP.java
-rw-r--r--  1 ronkoron     4.6K Dec 22  2011 HTTPHEADER.java
-rw-r--r--  1 ronkoron     1.5K Dec 22  2011 MC.java
-rw-r--r--  1 ronkoron     721B Dec 22  2011 Namespace.java
-rw-r--r--  1 ronkoron      16K May  5 15:43 OWL.java
-rw-r--r--  1 ronkoron     5.7K May  5 15:44 RDF.java
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 RDFS.java
-rw-r--r--  1 ronkoron     442B Dec 22  2011 RSS.java
-rw-r--r--  1 ronkoron     5.1K Dec 22  2011 SIOC.java
-rw-r--r--  1 ronkoron     390B Dec 22  2011 SKOS.java
```

```
-rw-r--r--  1 ronkoron     957B Dec 22  2011 VCARD.java
-rw-r--r--  1 ronkoron     169B Dec 22  2011 VOID.java
-rw-r--r--  1 ronkoron     239B Dec 22  2011 XHTML.java
-rw-r--r--  1 ronkoron     2.7K Dec 22  2011 XSD.java
-rw-r--r--  1 ronkoron     138B Dec 22  2011 YARS2.java

NxParser/org/semanticweb/yars/nx/parser:
-rw-r--r--  1 ronkoron     149B Dec 22  2011 BNodeHandler.java
-rw-r--r--  1 ronkoron     429B Apr 18 00:31 Callback.java
-rw-r--r--  1 ronkoron      11K Jan 21  2012 NqParser.java
-rw-r--r--  1 ronkoron      13K Jan 21  2012 NxParser.java
-rw-r--r--  1 ronkoron     331B Dec 22  2011 ParseException.java

NxParser/org/semanticweb/yars/nx/reorder:
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 ReorderIterator.java

NxParser/org/semanticweb/yars/nx/sort:
-rw-r--r--  1 ronkoron     5.0K May 20 18:09 MergeSortIterator.java
-rw-r--r--  1 ronkoron     8.3K Dec 22  2011 SortIterator.java

NxParser/org/semanticweb/yars/nx/util:
-rw-r--r--  1 ronkoron     7.2K Dec 22  2011 NxUtil.java

NxParser/org/semanticweb/yars/stats:
-rw-r--r--  1 ronkoron     222B Dec 22  2011 Analyser.java
-rw-r--r--  1 ronkoron     5.4K Dec 22  2011 Count.java
-rw-r--r--  1 ronkoron     1.5K Dec 22  2011 CountNodeTypeAnalyser.java
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 CountNodeTypeHashAnalyser.java
-rw-r--r--  1 ronkoron     691B Dec 22  2011 CountStmtAnalyser.java
-rw-r--r--  1 ronkoron     1.7K Dec 22  2011 DefaultAnalyser.java
-rw-r--r--  1 ronkoron     1.1K Dec 22  2011 DistributionAnalyser.java
-rw-r--r--  1 ronkoron     1.2K Dec 22  2011 DistributionArrayAnalyser.java
-rw-r--r--  1 ronkoron     538B Dec 22  2011 InputAnalyser.java
-rw-r--r--  1 ronkoron     1.4K Dec 22  2011 NodeTransformer.java
-rw-r--r--  1 ronkoron     3.6K Dec 22  2011 Statistics.java
-rw-r--r--  1 ronkoron     3.3K Dec 22  2011 VoiD.java
drwxr-xr-x  4 ronkoron     136B May 13 14:38 output

NxParser/org/semanticweb/yars/stats/output:
-rw-r--r--  1 ronkoron     2.6K Dec 22  2011 ToVoid.java

NxParser/org/semanticweb/yars/tld:
-rw-r--r--  1 ronkoron     2.5K Dec 22  2011 Tld.java
-rw-r--r--  1 ronkoron     7.5K Dec 22  2011 TldManager.java
-rw-r--r--  1 ronkoron     8.0K Dec 22  2011 URIHandler.java
```

```
-rw-r--r--  1 ronkoron      38K Dec 22  2011 tld.dat

NxParser/org/semanticweb/yars/util:
-rw-r--r--   1 ronkoron     1.9K Dec 22  2011 Array.java
-rw-r--r--   1 ronkoron     938B Apr 19 23:21 CallbackBlockingQueue.java
-rw-r--r--   1 ronkoron     965B Apr  8 14:51 CallbackContextSet.java
-rw-r--r--   1 ronkoron     681B Apr 19 23:22 CallbackCount.java
-rw-r--r--   1 ronkoron     276B Apr  8 15:00 CallbackNQOutputStream.java
-rw-r--r--   1 ronkoron     1.8K Apr 19 23:22 CallbackNxBufferedWriter.java
-rw-r--r--   1 ronkoron     2.1K Apr 19 23:23 CallbackNxOutputStream.java
-rw-r--r--   1 ronkoron     3.5K Apr  8 14:51 CallbackRDFXMLOutputStream.java
-rw-r--r--   1 ronkoron     778B Apr  8 14:51 CallbackSet.java
-rw-r--r--   1 ronkoron     2.7K Apr 19 23:23 CallbackTicks.java
-rw-r--r--   1 ronkoron     753B Apr 19 23:21 Callbacks.java
-rw-r--r--   1 ronkoron     1.9K Dec 22  2011 CheckLengthIterator.java
-rw-r--r--   1 ronkoron     2.7K Dec 22  2011 CheckSortedIterator.java
-rw-r--r--   1 ronkoron     963B Dec 22  2011 FlyweightNodeIterator.java
-rw-r--r--   1 ronkoron     741B Dec 22  2011 LRUMapCache.java
-rw-r--r--   1 ronkoron     1.3K Dec 22  2011 LRUSetCache.java
-rw-r--r--   1 ronkoron     867B Dec 22  2011 PleaseCloseTheDoorWhenYouLeave
                                              Iterator.java
-rw-r--r--   1 ronkoron     270B Dec 22  2011 ResetableIterator.java
-rw-r--r--   1 ronkoron     976B Dec 22  2011 SideCallbackIterator.java
-rw-r--r--   1 ronkoron     843B Dec 22  2011 SniffIterator.java
-rw-r--r--   1 ronkoron     2.5K Dec 22  2011 TicksIterator.java
drwxr-xr-x  12 ronkoron     408B May 13 14:38 thread

NxParser/org/semanticweb/yars/util/thread:
-rw-r--r--  1 ronkoron     921B Dec 22  2011 BlockingQueueIterator.java
-rw-r--r--  1 ronkoron     779B Apr  8 14:51 CallbackBlockingQueue.java
-rw-r--r--  1 ronkoron     1.6K Dec 22  2011 ConsumerProducerThread.java
-rw-r--r--  1 ronkoron     1.7K Dec 22  2011 ConsumerThread.java
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 ProducerThread.java

NxParser/org/semanticweb/yars2:
drwxr-xr-x  17 ronkoron     578B May 26 18:45 rdfxml

NxParser/org/semanticweb/yars2/rdfxml:
-rw-r--r--  1 ronkoron     1.3K Dec 22  2011 ParserThread.java
-rw-r--r--  1 ronkoron     499B Apr  8 14:49 PrintCallBack.java
-rw-r--r--  1 ronkoron      12K May 26 18:45 RDFXMLParser.java
-rw-r--r--  1 ronkoron      41K May 22 21:20 RDFXMLParserBase.java
-rw-r--r--  1 ronkoron     5.6K Dec 22  2011 XMLRegex.java
drwxr-xr-x  3 ronkoron     102B May 26 18:45 org
```

```
LDspider/svn/tags/release-1.1/lib:
-rw-r--r--@  1 ronkoron      40K Oct 16  2009 commons-cli-1.2.jar
-rw-r--r--@  1 ronkoron     284K Dec 23  2009 httpclient-4.0.1.jar
-rw-r--r--@  1 ronkoron     169K Oct 16  2009 httpcore-4.0.1.jar
-rw-r--r--@  1 ronkoron     176K Oct 16  2009 httpcore-nio-4.0.1.jar
-rw-r--r--@  1 ronkoron      25K Dec 23  2009 httpmime-4.0.1.jar
-rw-r--r--@  1 ronkoron     712K Jun 10  2010 nxparser.jar

LDspider/svn/tags/release-1.1/dist:
-rw-r--r--@ 1 ronkoron      84K Sep  7  2010 ldspider-1.1-lib.jar
-rw-r--r--@ 1 ronkoron     1.4M Sep  7  2010 ldspider-1.1.jar
```

Appendix B

Seed Documents Used for Evaluation

http://www.ebusiness-unibw.org/ontologies/consumerelectronics/v1.owl
http://www.semanticbible.com/2004/04/NTNames.owl
http://www.r4isstatic.com/linkeddata/ontologies/football/football.owl
http://www.openmath.org/ontology/openmath.owl
http://openprovenance.org/model/opm.owl
http://www.somweb.se/ontologies/diagnosisOntology.owl
http://zaltys.net/ontology/AKTiveSAOntology.owl
http://www.fadyart.com/Finance.owl
http://motools.sourceforge.net/event/event.rdf
http://labs.mondeca.com/dataset/lov/agg/down/file_coo.rdf
http://www.ontologyportal.org/translations/SUMO.owl
http://ontorule-project.eu/resources/steel.owl
http://www.topbraid.org/2007/02/realEstate.owl
http://www.wsmo.org/TR/d32/v1.0/wsml.rdf
http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
http://motools.sourceforge.net/doc/musicontology.rdfs
http://opendatacommunities.org/datasets.rdf
http://labs.mondeca.com/vocabuse/lol.rdf
http://data.nytimes.com/organizations.rdf
http://schemapedia.com/schemas/biol.rdf

Appendix C

Vocabulary Statistics Output

```
Total vocabularies found: 17
Total vocabulary terms found: 477515

          Known Vocabulary Statistics
          ---------------------------

     dc  (8 terms):
          contributor :  1
          creator :  248
          date :  1
          description :  5
          rights :  106
          source :  2
          subject :  2
          title :  7
          =========================
          subtotal: 372

     dcterms  (1 terms):
          license :  1
          ========================
          subtotal: 1

     foaf  (2 terms):
          maker :  1
          Person :  1
          ========================
          subtotal: 2

     owl  (40 terms):
          AllDifferent :  12
```

```
     AllDisjointClasses :  14
     allValuesFrom :  170
     AnnotationProperty :  17
     cardinality :  50
     Class :  56085
     complementOf :  4
     DatatypeProperty :  184
     differentFrom :  3
     disjointObjectProperties :  1
     disjointWith :  99965
     distinctMembers :  112
     equivalentClass :  247
     equivalentProperty :  18
     FunctionalProperty :  78
     hasValue :  533
     imports :  17
     incompatibleWith :  1
     intersectionOf :  131
     InverseFunctionalProperty :  9
     inverseOf :  259
     maxCardinality :  23
     members :  33
     minCardinality :  5
     ObjectProperty :  1763
     onClass :  3
     oneOf :  6
     onProperty :  1288
     Ontology :  17
     OntologyProperty :  1
     priorVersion :  1
     propertyChainAxiom :  7
     propertyDisjointWith :  4
     Restriction :  1266
     sameAs :  6
     someValuesFrom :  507
     SymmetricProperty :  11
     TransitiveProperty :  99
     unionOf :  202
     versionInfo :  16
     ========================
     subtotal: 163168

rdf  (12 terms):
     about :  59054
     datatype :  14603
```

```
     Description :  2070
     first :  65
     ID :  5693
     nodeID :  201
     parseType :  469
     Property :  33
     RDF :  20
     resource :  58438
     rest :  65
     type :  734
     ========================
     subtotal: 141445

rdfs  (9 terms):
     Class :  19
     comment :  3622
     domain :  720
     isDefinedBy :  420
     label :  4088
     range :  851
     seeAlso :  172
     subClassOf :  4955
     subPropertyOf :  594
     ========================
     subtotal: 15441



     Unknown Vocabulary Statistics
     -----------------------------

dct  (5 terms):
     contributor :  5
     created :  4
     creator :  2
     subject :  2
     title :  2
     ========================
     subtotal: 15

gr  (2 terms):
     BusinessEntity :  156
     legalName :  156
     ========================
     subtotal: 312
```

```
j.0  (11 terms):
     hasOrigColumnName :  2
     ISICCode :  766
     ISICDescription :  766
     miccity :  595
     miccode :  595
     miccountry :  595
     micdate :  595
     micinstitution :  595
     micisocountrycode :  595
     micstatus :  595
     micwebsite :  595
     =========================
     subtotal: 6294

parties  (1 terms):
     Organisation :  1
     =========================
     subtotal: 1

rel  (2 terms):
     childOf :  1
     parentOf :  1
     =========================
     subtotal: 2

somweb  (14 terms):
     Afte :  2
     Cancer :  1
     Cand.ass :  2
     Candidos :  2
     Diag-def :  98
     Diag-tent :  106
     Dis-past :  2
     icdCode :  100
     LCR :  1
     LG :  3
     Lichen :  1
     OLPeu :  3
     OLPrpp :  3
     Snusforandring :  1
     =========================
     subtotal: 325

sp  (3 terms):
```

```
     Select :  1
     Variable :  6
     varName :  6
     ========================
     subtotal: 13

spin  (11 terms):
     abstract :  7
     AskTemplate :  1
     constraint :  7
     constructor :  1
     ConstructTemplate :  1
     Function :  2
     MagicProperty :  1
     RuleProperty :  1
     SelectTemplate :  1
     Template :  2
     UpdateTemplate :  1
     ========================
     subtotal: 25

spl  (6 terms):
     Attribute :  6
     ConstructDefaultValues :  1
     maxCount :  6
     minCount :  6
     predicate :  6
     valueType :  4
     ========================
     subtotal: 29

suo  (19 terms):
     BeliefGroup :  3
     Character :  2
     City :  86
     EthnicGroup :  14
     FreshWaterArea :  5
     GeopoliticalArea :  2
     GroupOfPeople :  4
     Island :  9
     LandArea :  8
     Man :  300
     Nation :  1
     NaturalLanguage :  5
     PoliticalOrganization :  6
```

```
     Region :  9
     ReligiousOrganization :  2
     SaltWaterArea :  3
     Serial :  2
     StateOrProvince :  22
     Woman :  42
     ========================
     subtotal: 525

vs  (1 terms):
     term_status :  21
     ========================
     subtotal: 21
```

Appendix D

Document Statistics Output

```
Number of documents crawled: 20
Number of OWL documents crawled: 18
Number of RDF documents crawled: 2
Number of Other documents crawled: 0

          Document Statistics
          -------------------

http://labs.mondeca.com/dataset/lov/agg/down/file_coo.rdf (OWL) : 1
   (outgoing edges: 4)       ()
     dc  (5 terms):
          contributor :  1
          creator :  1
          rights :  1
          subject :  1
          title :  1
          =========================
          subtotal: 5

     dcterms  (1 terms):
          license :  1
          =========================
          subtotal: 1

     owl  (4 terms):
          disjointWith :  72
          imports :  1
          unionOf :  2
          versionInfo :  1
          =========================
          subtotal: 76
```

```
      rdf  (8 terms):
           about :  47
           Description :  53
           first :  4
           nodeID :  12
           RDF :  1
           resource :  192
           rest :  4
           type :  38
           ========================
           subtotal: 351

      rdfs  (8 terms):
           comment :  30
           domain :  19
           isDefinedBy :  29
           label :  30
           range :  19
           seeAlso :  1
           subClassOf :  7
           subPropertyOf :  1
           ========================
           subtotal: 136

http://motools.sourceforge.net/event/event.rdf (OWL) : 1
   (outgoing edges: 3)       ()
     dc  (3 terms):
           date :  1
           description :  1
           title :  1
           ========================
           subtotal: 3

      foaf  (2 terms):
           maker :  1
           Person :  1
           ========================
           subtotal: 2

      owl  (14 terms):
           AnnotationProperty :  1
           Class :  6
           DatatypeProperty :  5
           disjointWith :  1
           equivalentClass :  2
```

```
          equivalentProperty :  8
          imports :  1
          inverseOf :  3
          ObjectProperty :  17
          onProperty :  2
          Ontology :  1
          Restriction :  2
          someValuesFrom :  2
          versionInfo :  1
          =========================
          subtotal: 52

     rdf  (4 terms):
          about :  31
          RDF :  1
          resource :  30
          type :  1
          =========================
          subtotal: 63

     rdfs  (4 terms):
          comment :  10
          domain :  7
          label :  14
          range :  4
          =========================
          subtotal: 35

     vs  (1 terms):
          term_status :  21
          =========================
          subtotal: 21

http://omdoc.org/ontology (OWL) : 1
   (outgoing edges: 1)       (http://www.openmath.org/ontology/openmath.owl)
     owl  (11 terms):
          differentFrom :  3
          disjointWith :  79
          hasValue :  2
          intersectionOf :  1
          inverseOf :  16
          onProperty :  4
          propertyChainAxiom :  5
          propertyDisjointWith :  4
          someValuesFrom :  2
```

```
          unionOf :  6
          versionInfo :  1
          ========================
          subtotal: 123

     rdf  (8 terms):
          about :  131
          Description :  174
          first :  32
          nodeID :  86
          RDF :  1
          resource :  580
          rest :  32
          type :  149
          ========================
          subtotal: 1185

     rdfs  (7 terms):
          comment :  103
          domain :  27
          isDefinedBy :  121
          label :  330
          range :  33
          subClassOf :  80
          subPropertyOf :  27
          ========================
          subtotal: 721

     dct  (4 terms):
          contributor :  5
          creator :  1
          subject :  1
          title :  1
          ========================
          subtotal: 8

http://openprovenance.org/model/opm.owl (OWL) : 1
   (outgoing edges: 1)       ()
     owl  (9 terms):
          allValuesFrom :  12
          Class :  40
          DatatypeProperty :  3
          members :  2
          ObjectProperty :  25
          onProperty :  12
```

```
          Ontology :  1
          Restriction :  12
          unionOf :  5
          ========================
          subtotal: 112

     rdf  (8 terms):
          about :  28
          datatype :  9
          Description :  10
          ID :  46
          parseType :  7
          RDF :  1
          resource :  103
          type :  15
          ========================
          subtotal: 219

     rdfs  (5 terms):
          comment :  9
          domain :  26
          range :  23
          subClassOf :  31
          subPropertyOf :  2
          ========================
          subtotal: 91

http://spinrdf.org/spin (RDF) : 1
   (outgoing edges: 0)       (http://www.fadyart.com/Finance.owl)
     owl  (4 terms):
          imports :  1
          Ontology :  1
          OntologyProperty :  1
          versionInfo :  1
          ========================
          subtotal: 4

     rdf  (7 terms):
          about :  24
          datatype :  104
          Description :  5
          ID :  48
          Property :  33
          RDF :  1
          resource :  68
```

```
     ========================
     subtotal: 283

rdfs  (7 terms):
     Class :  19
     comment :  50
     domain :  16
     label :  44
     range :  19
     subClassOf :  25
     subPropertyOf :  20
     ========================
     subtotal: 193

sp  (2 terms):
     Variable :  6
     varName :  6
     ========================
     subtotal: 12

spin  (11 terms):
     abstract :  7
     AskTemplate :  1
     constraint :  6
     constructor :  1
     ConstructTemplate :  1
     Function :  2
     MagicProperty :  1
     RuleProperty :  1
     SelectTemplate :  1
     Template :  1
     UpdateTemplate :  1
     ========================
     subtotal: 23

spl  (6 terms):
     Attribute :  6
     ConstructDefaultValues :  1
     maxCount :  6
     minCount :  6
     predicate :  6
     valueType :  4
     ========================
     subtotal: 29
```

```
http://topbraid.org/sxml (RDF) : 1
   (outgoing edges: 1)       (http://www.fadyart.com/Finance.owl)
     owl  (7 terms):
          AnnotationProperty :  2
          Class :  8
          DatatypeProperty :  1
          imports :  1
          ObjectProperty :  1
          Ontology :  1
          versionInfo :  1
          ========================
          subtotal: 15

     rdf  (5 terms):
          about :  4
          datatype :  4
          ID :  9
          RDF :  1
          resource :  9
          ========================
          subtotal: 27

     rdfs  (6 terms):
          comment :  5
          domain :  4
          label :  8
          range :  1
          subClassOf :  5
          subPropertyOf :  1
          ========================
          subtotal: 24

http://www.ebusiness-unibw.org/ontologies/consumerelectronics/v1.owl
   (OWL) : 2   (outgoing edges: 152)       ()
     dc  (4 terms):
          creator :  1
          rights :  1
          subject :  1
          title :  1
          ========================
          subtotal: 4

     owl  (6 terms):
          AnnotationProperty :  6
          Class :  49
```

```
          DatatypeProperty :  28
          imports :  1
          ObjectProperty :  122
          Ontology :  1
          ========================
          subtotal: 207

     rdf  (5 terms):
          about :  6
          datatype :  324
          ID :  641
          RDF :  1
          resource :  703
          ========================
          subtotal: 1675

     rdfs  (8 terms):
          comment :  642
          domain :  1
          isDefinedBy :  199
          label :  1125
          range :  149
          seeAlso :  154
          subClassOf :  49
          subPropertyOf :  150
          ========================
          subtotal: 2469

     gr  (2 terms):
          BusinessEntity :  156
          legalName :  156
          ========================
          subtotal: 312

http://www.fadyart.com/Finance.owl (OWL) : 1  (outgoing edges: 11)       ()
     dc  (2 terms):
          rights :  2
          source :  1
          ========================
          subtotal: 3

     owl  (22 terms):
          AllDisjointClasses :  14
          allValuesFrom :  15
          AnnotationProperty :  3
```

```
     Class :  492
     DatatypeProperty :  7
     disjointWith :  142
     equivalentClass :  25
     FunctionalProperty :  5
     imports :  7
     incompatibleWith :  1
     intersectionOf :  15
     InverseFunctionalProperty :  2
     inverseOf :  2
     members :  14
     ObjectProperty :  101
     onProperty :  49
     Ontology :  1
     priorVersion :  1
     Restriction :  49
     someValuesFrom :  34
     unionOf :  21
     versionInfo :  1
     ========================
     subtotal: 1001

rdf  (8 terms):
     about :  412
     datatype :  104
     Description :  28
     ID :  318
     parseType :  51
     RDF :  1
     resource :  380
     type :  7
     ========================
     subtotal: 1301

rdfs  (8 terms):
     comment :  229
     domain :  40
     isDefinedBy :  1
     label :  233
     range :  30
     seeAlso :  6
     subClassOf :  193
     subPropertyOf :  33
     ========================
     subtotal: 765
```

```
     parties  (1 terms):
          Organisation :  1
          ========================
          subtotal: 1

     sp  (1 terms):
          Select :  1
          ========================
          subtotal: 1

     spin  (2 terms):
          constraint :  1
          Template :  1
          ========================
          subtotal: 2

http://www.fadyart.com/iso20022.owl (OWL) : 1
   (outgoing edges: 1)       (http://www.fadyart.com/Finance.owl)
     dc  (1 terms):
          rights :  102
          ========================
          subtotal: 102

     owl  (4 terms):
          Class :  769
          disjointWith :  1192
          imports :  1
          Ontology :  1
          ========================
          subtotal: 1963

     rdf  (6 terms):
          about :  671
          datatype :  49
          Description :  25
          ID :  124
          RDF :  1
          resource :  695
          ========================
          subtotal: 1565

     rdfs  (4 terms):
          comment :  82
          isDefinedBy :  3
          label :  123
```

```
          subClassOf :  147
          ========================
          subtotal: 355

http://www.fadyart.com/ontologies/data/Finance.owl (OWL) : 1
   (outgoing edges: 2)       (http://www.fadyart.com/Finance.owl)
     dc  (1 terms):
          source :  1
          ========================
          subtotal: 1

     owl  (21 terms):
          allValuesFrom :  42
          AnnotationProperty :  4
          Class :  429
          DatatypeProperty :  74
          disjointObjectProperties :  1
          disjointWith :  332
          distinctMembers :  100
          equivalentClass :  54
          intersectionOf :  30
          inverseOf :  90
          maxCardinality :  1
          members :  17
          minCardinality :  2
          ObjectProperty :  247
          onClass :  3
          onProperty :  131
          Ontology :  1
          Restriction :  131
          someValuesFrom :  86
          unionOf :  76
          versionInfo :  1
          ========================
          subtotal: 1852

     rdf  (7 terms):
          about :  4649
          datatype :  11778
          Description :  1651
          parseType :  223
          RDF :  1
          resource :  2380
          type :  179
          ========================
```

```
          subtotal: 20861

     rdfs  (8 terms):
          comment :  925
          domain :  317
          isDefinedBy :  3
          label :  413
          range :  292
          seeAlso :  10
          subClassOf :  368
          subPropertyOf :  295
          =========================
          subtotal: 2623

     j.0  (11 terms):
          hasOrigColumnName :  2
          ISICCode :  766
          ISICDescription :  766
          miccity :  595
          miccode :  595
          miccountry :  595
          micdate :  595
          micinstitution :  595
          micisocountrycode :  595
          micstatus :  595
          micwebsite :  595
          =========================
          subtotal: 6294

http://www.openmath.org/ontology/openmath.owl (OWL) : 1
   (outgoing edges: 2)       ()
     owl  (12 terms):
          allValuesFrom :  6
          cardinality :  10
          disjointWith :  58
          imports :  1
          inverseOf :  7
          maxCardinality :  1
          minCardinality :  1
          oneOf :  2
          onProperty :  18
          propertyChainAxiom :  2
          unionOf :  5
          versionInfo :  1
          =========================
```

```
          subtotal: 112

     rdf  (9 terms):
          about :  72
          datatype :  12
          Description :  124
          first :  29
          nodeID :  103
          RDF :  1
          resource :  385
          rest :  29
          type :  114
          ========================
          subtotal: 869

     rdfs  (7 terms):
          comment :  62
          domain :  26
          isDefinedBy :  64
          label :  63
          range :  25
          subClassOf :  36
          subPropertyOf :  14
          ========================
          subtotal: 290

     dct  (3 terms):
          creator :  1
          subject :  1
          title :  1
          ========================
          subtotal: 3

http://www.r4isstatic.com/linkeddata/ontologies/football/football.owl
   (OWL) : 1   (outgoing edges: 3)       ()
     dc  (3 terms):
          creator :  1
          description :  1
          title :  1
          ========================
          subtotal: 3

     owl  (6 terms):
          Class :  33
          inverseOf :  14
```

```
          ObjectProperty :  27
          Ontology :  1
          unionOf :  1
          versionInfo :  1
          =========================
          subtotal: 77

     rdf  (5 terms):
          about :  5
          ID :  55
          parseType :  1
          RDF :  1
          resource :  97
          =========================
          subtotal: 159

     rdfs  (6 terms):
          comment :  29
          domain :  27
          label :  56
          range :  27
          subClassOf :  28
          subPropertyOf :  2
          =========================
          subtotal: 169

     dct  (1 terms):
          created :  1
          =========================
          subtotal: 1

http://www.r4isstatic.com/linkeddata/ontologies/ontomedia/core/
       expression.owl
   (OWL) : 1    (outgoing edges: 5)
   (http://www.r4isstatic.com/linkeddata/ontologies/football/football.owl)
     dc  (3 terms):
          creator :  3
          description :  1
          title :  1
          =========================
          subtotal: 5

     owl  (7 terms):
          Class :  37
          FunctionalProperty :  4
```

```
          inverseOf :  23
          ObjectProperty :  44
          Ontology :  1
          unionOf :  7
          versionInfo :  1
          ========================
          subtotal: 117

     rdf  (5 terms):
          about :  15
          ID :  64
          parseType :  7
          RDF :  1
          resource :  140
          ========================
          subtotal: 227

     rdfs  (6 terms):
          comment :  64
          domain :  48
          label :  63
          range :  48
          subClassOf :  16
          subPropertyOf :  12
          ========================
          subtotal: 251

     dct  (1 terms):
          created :  1
          ========================
          subtotal: 1

http://www.r4isstatic.com/linkeddata/ontologies/ontomedia/core/space.owl
   (OWL) : 1    (outgoing edges: 3)
   (http://www.r4isstatic.com/linkeddata/ontologies/football/football.owl)
     dc  (3 terms):
          creator :  3
          description :  1
          title :  1
          ========================
          subtotal: 5

     owl  (6 terms):
          Class :  57
          imports :  1
```

```
          inverseOf :  1
          ObjectProperty :  2
          Ontology :  1
          versionInfo :  1
          =========================
          subtotal: 63

     rdf  (4 terms):
          about :  2
          ID :  58
          RDF :  1
          resource :  64
          =========================
          subtotal: 125

     rdfs  (6 terms):
          comment :  59
          domain :  2
          label :  60
          range :  1
          subClassOf :  58
          subPropertyOf :  1
          =========================
          subtotal: 181

     dct  (1 terms):
          created :  1
          =========================
          subtotal: 1

http://www.r4isstatic.com/linkeddata/ontologies/ontomedia/ext/common/being.owl
   (OWL) : 1    (outgoing edges: 5)
   (http://www.r4isstatic.com/linkeddata/ontologies/football/football.owl)
     dc  (3 terms):
          creator :  3
          description :  1
          title :  1
          =========================
          subtotal: 5

     owl  (12 terms):
          allValuesFrom :  3
          Class :  166
          equivalentClass :  3
          FunctionalProperty :  1
```

```
        imports :  2
        inverseOf :  11
        ObjectProperty :  56
        onProperty :  3
        Ontology :  1
        Restriction :  3
        unionOf :  33
        versionInfo :  1
        =========================
        subtotal: 283

    rdf  (5 terms):
        about :  106
        ID :  85
        parseType :  33
        RDF :  1
        resource :  141
        =========================
        subtotal: 366

    rdfs  (6 terms):
        comment :  85
        domain :  57
        label :  86
        range :  57
        subClassOf :  29
        subPropertyOf :  12
        =========================
        subtotal: 326

    dct  (1 terms):
        created :  1
        =========================
        subtotal: 1

http://www.semanticbible.com/2004/04/NTNames.owl (OWL) : 1
   (outgoing edges: 3)       ()
    owl  (13 terms):
        allValuesFrom :  2
        Class :  91
        DatatypeProperty :  6
        InverseFunctionalProperty :  5
        inverseOf :  12
        ObjectProperty :  19
        onProperty :  2
```

```
     Ontology :  1
     Restriction :  2
     sameAs :  6
     SymmetricProperty :  4
     unionOf :  6
     versionInfo :  1
     =========================
     subtotal: 157

rdf  (6 terms):
     about :  83
     ID :  630
     parseType :  6
     RDF :  1
     resource :  102
     type :  17
     =========================
     subtotal: 839

rdfs  (5 terms):
     comment :  75
     domain :  25
     label :  633
     range :  25
     subClassOf :  48
     =========================
     subtotal: 806

rel  (2 terms):
     childOf :  1
     parentOf :  1
     =========================
     subtotal: 2

suo  (19 terms):
     BeliefGroup :  3
     Character :  2
     City :  86
     EthnicGroup :  14
     FreshWaterArea :  5
     GeopoliticalArea :  2
     GroupOfPeople :  4
     Island :  9
     LandArea :  8
     Man :  300
```

```
          Nation :  1
          NaturalLanguage :  5
          PoliticalOrganization :  6
          Region :  9
          ReligiousOrganization :  2
          SaltWaterArea :  3
          Serial :  2
          StateOrProvince :  22
          Woman :  42
          =========================
          subtotal: 525

http://www.somweb.se/ontologies/diagnosisOntology.owl (OWL) : 1
   (outgoing edges: 1)       ()
     dc  (1 terms):
          creator :  236
          =========================
          subtotal: 236

     owl  (4 terms):
          AnnotationProperty :  1
          Class :  16
          DatatypeProperty :  1
          Ontology :  1
          =========================
          subtotal: 19

     rdf  (4 terms):
          about :  244
          RDF :  1
          resource :  206
          type :  197
          =========================
          subtotal: 648

     rdfs  (2 terms):
          label :  234
          subClassOf :  9
          =========================
          subtotal: 243

     somweb  (14 terms):
          Afte :  2
          Cancer :  1
          Cand.ass :  2
```

```
          Candidos :  2
          Diag-def :  98
          Diag-tent :  106
          Dis-past :  2
          icdCode :  100
          LCR :  1
          LG :  3
          Lichen :  1
          OLPeu :  3
          OLPrpp :  3
          Snusforandring :  1
          ========================
          subtotal: 325

http://www.topbraid.org/2007/05/composite.owl (OWL) : 1
   (outgoing edges: 0)       (http://topbraid.org/sxml)
     owl  (5 terms):
          DatatypeProperty :  1
          inverseOf :  2
          ObjectProperty :  3
          Ontology :  1
          versionInfo :  1
          ========================
          subtotal: 8

     rdf  (5 terms):
          about :  2
          datatype :  1
          ID :  3
          RDF :  1
          resource :  2
          ========================
          subtotal: 9

     rdfs  (2 terms):
          comment :  4
          range :  1
          ========================
          subtotal: 5

http://www.w3.org/2006/time (OWL) : 1
   (outgoing edges: 1)
      (http://motools.sourceforge.net/event/event.rdf)
     owl  (13 terms):
          cardinality :  8
```

```
          Class :  14
          DatatypeProperty :  17
          disjointWith :  1
          hasValue :  2
          inverseOf :  7
          maxCardinality :  17
          ObjectProperty :  24
          oneOf :  2
          onProperty :  27
          Ontology :  1
          Restriction :  27
          unionOf :  1
          ========================
          subtotal: 148

     rdf  (7 terms):
          about :  18
          datatype :  26
          ID :  52
          parseType :  3
          RDF :  1
          resource :  112
          type :  1
          ========================
          subtotal: 213

     rdfs  (6 terms):
          comment :  1
          domain :  34
          range :  34
          seeAlso :  1
          subClassOf :  33
          subPropertyOf :  1
          ========================
          subtotal: 104

http://zaltys.net/ontology/AKTiveSAOntology.owl (OWL) : 1
   (outgoing edges: 1)       ()
     owl  (27 terms):
          AllDifferent :  12
          allValuesFrom :  90
          cardinality :  32
          Class :  53878
          complementOf :  4
          DatatypeProperty :  41
```

```
     disjointWith :  98088
     distinctMembers :  12
     equivalentClass :  163
     equivalentProperty :  10
     FunctionalProperty :  68
     hasValue :  529
     intersectionOf :  85
     InverseFunctionalProperty :  2
     inverseOf :  71
     maxCardinality :  4
     minCardinality :  2
     ObjectProperty :  1075
     oneOf :  2
     onProperty :  1040
     Ontology :  1
     Restriction :  1040
     someValuesFrom :  383
     SymmetricProperty :  7
     TransitiveProperty :  99
     unionOf :  39
     versionInfo :  2
     ==========================
     subtotal: 156779

rdf  (7 terms):
     about :  52504
     datatype :  2192
     ID :  3560
     parseType :  138
     RDF :  1
     resource :  52049
     type :  16
     ==========================
     subtotal: 110460

rdfs  (6 terms):
     comment :  1158
     domain :  44
     label :  573
     range :  63
     subClassOf :  3793
     subPropertyOf :  23
     ==========================
     subtotal: 5654
```

References

Berners-Lee, T., Hendler, J., and Lassila, O. 2001. The semantic web. *Scientific American 284,* 5 (May), 34–43.

Bizer, C., Heath, T., and Berners-Lee, T. 2009. Linked data – the story so far. *International Journal on Semantic Web and Information Systems 5,* 3, 1–22.

Cormode, G. and Krishnamurthy, B. 2008. Key differences between web 1.0 and web 2.0. *First Monday 13,* 6.

Gruber, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition 5,* 2, 199–220.

Gutiérrez, C., Hurtado, C. A., and Mendelzon, A. O. 2004. Foundations of semantic web databases. In *PODS*, C. Beeri and A. Deutsch, Eds. ACM, 95–106.

Heflin, J. 2004. Web ontology language (owl) use cases and requirements. World Wide Web Consortium, Recommendation REC-webont-req-20040210.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. 2009. Owl 2 web ontology language primer. W3C Recommendation. Available at `http://www.w3.org/TR/2009/REC-owl2-primer-20091027/`.

Isele, R., Umbrich, J., Bizer, C., and Harth, A. 2010. LDSpider: An open-source crawling framework for the web of linked data. In *Proceedings of 9th International Semantic Web Conference (ISWC 2010) Posters and Demos.*

Manola, F. and Miller, E. 2004. RDF primer. *W3C Recommendation 10*, 1–107.

McGuinness, D. L. and Van Harmelen, F. 2004. OWL Web Ontology Language overview. *W3C Recommendation 10*, 1–19.

Prud'hommeaux, E. and Seaborne, A. 2008. SPARQL query language for rdf. *W3C Recommendation 4*, 1–106.

W3C OWL Working Group. 2009. OWL 2 Web Ontology Language Document Overview.

Wright, R. 1997. The man who invented the web. *Time 149,* 20, 64.