ABSTRACT

Title of Dissertation: Enhancing Semantic Web Data Access

Li Ding, 2006

Dissertation directed by: Dr. Tim Finin
Professor
Department of Computer Science and Electrical Engineering

The Semantic Web was invented by Tim Berners-Lee in 1998 as a web of data for machine consumption. Its applicability in supporting real world applications on the World Wide Web, however, remains unclear to this day because most existing works treat the Semantic Web as one universal RDF graph and ignore the Web aspect. In fact, the Semantic Web is distributed on the Web as a web of belief: each piece of Semantic Web data is independently published on the Web as a certain agent's belief instead of the universal truth.

Therefore, we enhance the current conceptual model of the Semantic Web to characterize both the content and the context of Semantic Web data. A significant sample dataset is harvested to demonstrate the non-trivial presence and the global properties of the Semantic Web on the Web. Based on the enhanced conceptual model, we introduce a novel search and navigation model for the unique behaviors in Web-scale Semantic Web data access, and develop an enabling tool – the Swoogle Semantic Web search engine. To evaluate the data quality of Semantic Web data, we also (i) develop an explainable ranking schema that orders the popularity of Semantic Web documents and terms, and (ii) introduce a new level of granularity of Semantic Web data– RDF molecule that supports lossless RDF graph decomposition and effective provenance tracking.

This dissertation systematically investigates the Web aspect of the Semantic Web. Its primary contributions are the enhanced conceptual model of the Semantic Web, the novel Semantic Web search and navigation model, and the Swoogle Semantic Web search engine.

Enhancing Semantic Web Data Access

by
Li Ding

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

To my parents.

Acknowledgement

First and foremost, I would like to thank my advisor, Tim Finin. The dissertation would not have been possible without him. His guidance, understanding and support are inspiring and motivational.

I am thankful to Anupam Joshi for valuable suggestions and generous supports. I am also grateful to my other committee members: Yun Peng, James Mayfield, Filip Perich and Charles Nicolas, from whom I got insightful feedbacks on my work.

I would like to thank my former academic advisor, Zhuoqun Xu at Peking University, who brought me into the world of artificial intelligence.

I would like to thank Marie desJardins, Hillol Kargupta, Yelena Yesha, Lina Zhou, Dongsong Zhang, and other faculty members at UMBC. Marie desJardins gave me useful guidance for being a good researcher. The half-year collaboration with Hillol Kargupta strengthened my confidence in my research capability.

I would like to thank Rong Pan, Pavan Reddivari, Pranam Kolari, Akshay Java, Joel Sachs, Sandor Dornbush for joyful collaboration on Swoogle Project. I would also thank my colleagues who made my life a fun time: Zhongli Ding, Olga Ratsimor, Anand Patwardhan, Lalana Kagal, Harry Chen, Youyong Zou, and everybody else in eBiquity group. To all my other friends in the Department of CSEE at UMBC, thank you.

I would like to thank Amit Sheth, Boanerges Aleman-Meza and other members of LSDIS at University of Georgia for pleasant collaboration in 2005.

I would like to thank Deborah McGuiness, Richard Fikes, Paulo Pinheiro da Silva, Honglei Zeng, Cynthia Chang, Debbie Barros and other members of KSL at Stanford University, where I spent three wonderful months there evaluating my thesis work in 2005.

I would like to thank my host family Nancy Cutair, June Auer, and Michael Auer, for showing me the American culture and healing my homesick.

Special thanks must go to my parents, my brother, and my beloved fiancée Yongmei Shi. Their love, trust, patience and encouragement through the years towards my PhD degree are the most precious asset in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

INTRODUCTION

"Q: I'm updating my address book entries on (some site which shares contact information). Could you log on and update your address book, please? Then we can keep in touch and easily track changes to each other's addresses.

A: No, I have a FOAF file. Do you? Why should I have to get an account at every site which keeps a record of me? That's not using the web. In fact I have that information on the web as data. A URI for me is

`http://www.w3.org/People/Berners-Lee/card#i`

That is available in RDF, the W3C standard for generic data interchange, as card.rdf, ... If you are updating your address book, please take the time to publish a FOAF page."

– Tim Berners-Lee[1]

I.A The Semantic Web on the Web

"*The Semantic Web is a web of data, in some ways like a global database*" [10]. From the famous article written by Tim Berners-Lee et al. [13], we found two important aspects of the Semantic Web, namely the *Web aspect* and the *semantic aspect*. The **Web aspect** acknowledges the fact that the Semantic Web is part

[1] `http://www.w3.org/People/Berners-Lee/FAQ.html`

of the World Wide Web and emphasizes the applicability of Web-scale *Semantic Web data access*[2]. According to the Web aspect, Semantic Web data is distributed on the Web and accessible via HTTP (Hypertext Transfer Protocol) [53]; therefore, the Semantic Web is not merely a conventional database or knowledge-base whose data is managed in centralized manner. The **semantic aspect** acknowledges the rich semantics of knowledge[3] and emphasizes machine friendly knowledge representation. According to the semantic aspect, Semantic Web data is encoded and structured using Semantic Web languages such as RDF (Resource Description Framework) [95], RDFS (RDF Schema) [79] and OWL (Web Ontology Language) [35]; therefore, the Semantic Web is not merely the conventional Web in which semantics is hidden in plain text.

These two aspects together indicate that the Semantic Web is actually the *Semantic Web on the Web*, which can be thought of as a collection of loosely federated databases that share common logical knowledge representation, i.e., RDF graph model [95], but are physically published across the Web by independent owners. This vision is quite different from the well-known model that treats the Semantic Web as a universal RDF graph because it additionally considers how Semantic Web data is distributed on the Web.

It is notable that Semantic Web data on the Web is usually accessed in the form of a *Semantic Web document* (SWD), which is a Web document containing data encoded in Semantic Web languages. Semantic Web documents are used as the transfer packets in Web-scale Semantic Web data access, regardless of the underlying physical storage. They are not necessarily static files since they can be dynamically generated by Web services from database entries.

I.B Motivations

Most existing works simply assume that Semantic Web data is ready for access as a universal RDF graph. This assumption helps users to concentrate on the semantic aspect and hide the details of physical storage. It may be applicable when the desired data is managed by semantic storage systems such as RDF database systems [158, 21, 159] and peer-to-peer RDF storage systems [117, 72, 23]; however, it does not hold in the Semantic Web on the Web because the desired data must be obtained from the Web. In this dissertation, we are interested in Web-scale Semantic Web data access because it acknowledges the semantic aspect as well as the Web aspect. We have observed the practice of Web-scale Semantic Web data access in many real world applications, especially in the following scenarios:

[2] `http://www.w3.org/2001/sw/DataAccess/`

[3] *knowledge* and *data* are used interchangeably in this dissertation.

Scenario 1: finding Semantic Web ontologies – *Semantic Web ontologies* are Semantic Web documents that explicitly represent " *the meaning of terms in vocabularies and the relationships between those terms*" [35]. Like words in natural languages, *URI references* [12] are the basic *terms* in Semantic Web ontologies. Users search Semantic Web ontologies for various purposes: ontology engineers may find ontologies for reuse, extension, replacement or other purposes; and information publishers may find popular ontologies for publishing their data. For example, upon creating the Semantic Web Portal Ontology[4] [115], ontology engineers reuse terms from existing popular ontologies: the *person* concepts from FOAF (Friend-Of-A-Friend) ontology[5] [20], and the *News item* concept from RSS (RDF Site Summary) Ontology[6] [130]. Here, users search for relevant ontologies from the Web and choose the most popular ones to enhance the visibility and interoperability of the new ontology.

Scenario 2: enumerating inverse links of *owl:imports* – Semantic Web ontologies are interlinked but they evolve independently. Prior to updating an existing ontology, ontology engineers need to (i) enumerate the ontologies importing the present ontology and then (ii) evaluate the potential negative impacts on those ontologies. For example, the protege ontology[7] is imported by 38 ontologies and it has been updated more than four times during 2005[8]. It is notable that computing such inverse links requires a global catalog.

Scenario 3: enumerating all instance documents – Semantic Web based applications often treat the Semantic Web as a database and need all Semantic Web documents containing instances of a specified class or using a specified namespace. For example, FOAF personal profile documents have been found in large number [44], and they have been used as databases in award-wining Semantic Web based applications such as Seco [77] and Flink [112].

Scenario 4: finding popular instance properties of a class – Semantic Web languages offer a series of ontology constructs for describing "*groups of related resources and the relationships between these resources*" in terms of *class* and *property* [19]. RDF draws ideas from *frame-based systems* [113] and enables users to describe *facets* (i.e. instance property) of class-instances. An **instance property** p of a class c can be directly defined by a triple[9] (p, *rdfs:domain*, c) in Semantic Web ontologies or inductively

[4] `http://sw-portal.deri.at/ontologies/swportal`
[5] `http://xmlns.com/foaf/0.1/`
[6] `http://purl.org/rss/1.0/schema.rdf`
[7] `http://protege.stanford.edu/plugins/owl/protege`
[8] The results is collected from http://swoogle.umbc.edu, as of March 31, 2006.
[9] *triple* refers to *RDF triple* which is defined in [95]. They are used interchangeably in this dissertation.

learned from the instantiation of such *rdfs:domain* definitions in c's instances. For example, although not explicitly defined in ontologies, *rdfs:seeAlso* is used as a popular instance-property of *foaf:Person* because it has been instantiated by millions of class-instances of *foaf:Person*. Indeed, the instance properties learned from instance data are good sources for enhancing the domain definition in existing ontologies such as Dublin Core element ontology[10].

Scenario 5: finding (partial) evidences in the Semantic Web – An RDF graph is interpreted as making assertions about a set of resources. Suppose my FOAF document has asserted (i) the binding between an email "ding.li@umbc.edu" and a name "Li Ding" and (ii) information about my friends. Although the entire RDF graph in my FOAF document[11] may not be supported by any online FOAF document, its sub-graph may be supported. These (partial) evidences together enable us to compute the trustworthiness of my FOAF document using provenance-based methods [57, 136, 62, 42].

The above scenarios illustrate the requirements of Web-scale Semantic Web data access in real world applications: (i) a global catalog is required by all scenarios but it is hard to obtain due to the open architecture of the Web and the sparse distribution of Semantic Web data on the Web; and (ii) Semantic Web surfing is not merely following hyper links because data may be accessed at various levels of granularity with different constraints. Unfortunately, existing works either circumvent the Web aspect or provide inadequate support to Web-scale Semantic Web data access.

- *Semantic storage systems* cannot replace the Web because they require either centralized data management or homogenous data access protocol beyond HTTP.
- Conventional Web search engines such as Google and Yahoo are not effective in differentiating Semantic Web documents from the haystack of conventional Web documents because they treat semantic markups as plain text.
- *Semantic annotation systems* [105, 141, 73] annotate and search Web documents with structured metadata, but they are not specialized for the semantic content and structure of Semantic Web documents.
- Other systems specialized for the Semantic Web, such as ontology library systems [83, 85], RDF crawlers (aka. scutter) [127, 34, 18, 16], and Semantic Web databanks [86, 84], are limited in their coverage over the online Semantic Web data and their support to Semantic Web surfing.

[10] `http://purl.org/dc/elements/1.1/`
[11] Semantic Web document that instantiate classes in FOAF ontology.

I.C Objectives

In order to capture the Web aspect of the Semantic Web and enhance Web-scale Semantic Web data access, we should enhance the current conceptualization of the Semantic Web, build a global catalog of Semantic Web data on the Web, and design effective enabling tools and mechanisms. In this dissertation, we systematically investigate the following three critical research issues:

Modeling the Semantic Web from the Web aspect. The current approach that models the Semantic Web as one universal RDF graph is insufficient to capture the Web aspect because it does not model the context of the Semantic Web, such as the Web location and the agent creator of an RDF graph. Therefore, an enhanced conceptual model is needed to capture both the content and the context of the Semantic Web especially *knowledge provenance* [22, 31, 54] metadata. Moreover, a Semantic Web ontology for this model is needed to publish explicit metadata of the Semantic Web and make the Semantic Web self-descriptive.

Characterizing the Semantic Web. Effective harvesting methods are needed to obtain a significant dataset that can be used as a global catalog of the Semantic Web. In particular, we emphasize the diversity of the dataset, i.e., discovering more data sources (e.g., websites) and usage patterns (e.g., class and property usage) before downloading huge amount of samples from several giant data sources. Moreover, statistical measurements are needed to expose important (global) properties of the Semantic Web. In particular, the measurements should acknowledge the Web aspect and be substantial enough in guiding improvement and promoting adoption of Web-scale Semantic Web data access.

Enabling Web-scale Semantic Web data access. In order to effectively utilize the non-trivial amount of Semantic Web data available on the Web, two critical issues must be addressed: the *accessibility* issue, i.e., how to facilitate users accessing the desired Semantic Web data on the Web, and the *data quality* issue, i.e., how to measure *data quality* of Semantic Web data.

- For the *accessibility* issue, we need a conceptual model that characterizes the behaviors of Web-scale Semantic Web data access and effective tools that enable the conceptual model. A search engine based approach has been chosen due to its great success in addressing accessibility issues on the Web [100]. For example, search engines are such an important part of our Web surfing experience that the American Dialect Society chose the verb "*google*" unanimously as the "*most useful word of 2002*"[12]. The first four motivating scenarios previously listed above need a Semantic Web search engine that maintains a global catalog of the Semantic Web.

[12] `http://www.americandialect.org/index.php/amerdial/2003/01/`

- For the *data quality* issue, we mainly focus on provenance based approaches. Since "*Anyone Can Make Statements About Any Resource*" [95], "*Partial information is tolerated*" and "*there is no need for absolute truth*" [97], we cannot assume that all triples in an RDF graph are of high quality or that all Semantic Web data sources are of high quality. Therefore, we need effective mechanisms for tracking knowledge provenance of Semantic Web data at an appropriate level of granularity. The fifth motivating scenario requires such mechanisms.

I.D Contributions

This dissertation systematically addresses the Web aspect of the Semantic Web that is critical to the success of the Semantic Web in the real world but is always neglected in existing research.

The main theoretical contribution lies in the enhancement to the current conceptual model of the Semantic Web: we build the Web Of Belief (WOB) ontology that models both the content and the context of the Semantic Web, and a novel search and navigation model that characterizes surfing behaviors in Web-scale Semantic Web data access. Consequently, we derive an explainable ranking schema for ordering the popularity of Semantic Web documents and terms.

The main practical contribution is the Swoogle [40, 43] Semantic Web search engine, which provides a comprehensive support to the Semantic Web search and navigation model. The harvested dataset and corresponding measurements are also important contributions because they offer the global catalog and global properties that have been long desired by Semantic Web researchers and developers.

We additionally recognize the important role of RDF molecule [41] in decomposing and grouping RDF knowledge and investigate implementation issues in tracking knowledge provenance on the Semantic Web, especially in finding (partial) evidences.

I.E Thesis Statement

The Web aspect distinguishes the Semantic Web from previous works on knowledge representation and knowledge management. The Web Of Belief ontology and the search and navigation model are effective in modeling the Web aspect of the Semantic Web. The Swoogle Semantic Web search engine and its applications effectively demonstrate the applicability and importance of the conceptual models in Web-scale Semantic Web data access.

Chapter II

RESEARCH PROBLEMS AND RELATED WORKS

The Semantic Web is emerging on the Web as a web of data voluntarily contributed by many sources such as individual persons, websites, and web service agents [14]. It is a *web of data* because Semantic Web data is distributed on the Web and is encoded in machine friendly Semantic Web languages. In order to capture the Web aspect exhibited in the Semantic Web on the Web and fulfill the objectives mentioned in Chapter I, we systematically study five specific issues.

- *Modeling the Semantic Web* – The current model of the Semantic Web is insufficient for modeling the Web and agent context of the Semantic Web, so appropriate enhancement is needed.
- *Harvesting the Semantic Web* – We lack a global catalog, i.e. a significant sample dataset, of the real world Semantic Web, so effective harvesting methods are needed.
- *Measuring the Semantic Web* – We are ignorant of the deployment status and the global properties of the Semantic Web on the Web, so effective measurements are needed.
- *Surfing the Semantic Web* – Web-scale Semantic Web data access is different from Web surfing and database query, so it should be specially modeled and supported.
- *Provenance of Semantic Web Knowledge* – Tracking knowledge provenance of RDF graph on the Semantic Web should not miss relevant sources or introduce irrelevant sources, so appropriate level of granularity is needed.

II.A Modeling the Semantic Web and its Context

"*The Semantic Web is not a separate Web but an extension of the current one, ... Like the Internet, the Semantic Web will be as decentralized as possible ... the Web had to throw away the ideal of total consistency of all of its interconnections, ushering in the infamous message 'Error 404: Not Found' but allowing unchecked exponential growth*" [13]. The Web-based distributed publishing mechanism grants the Semantic Web a fast data growth model; however, it also requires users knowing the URL of Semantic Web data and accessing Semantic Web data using HTTP. Therefore, the Web serves as the storage layer of the Semantic Web and cannot be substituted by any existing *semantic storage system* in literature [7, 116, 101, 122] mainly because the Semantic Web on the Web is too huge and too dynamic.

- RDF database systems (aka. triple stores) always have scalability issues due to their centralized architecture, e.g. RSSDB [4], Jena [158], Kowari [159], Sesame [21], 3Store [76], and DLDB [68].
- Peer-to-peer RDF storage systems leverage scalability issues but require additional data access protocols beyond HTTP, e.g. Edutella [117], PeerDB [121], Piazza [72], RDFGrowth [152], MoGATU [129] and RDFPeers [23].
- Conventional storages wrapped by RDF converter have the same scalability problem as RDF database, e.g. knowledge base (KB) wrapper TAP [63], and database wrapper D2RQ [17].

The current model of the Semantic Web (i.e., one universal logical RDF graph), which has been adopted by most works in literature [10, 38, 81, 104, 52, 57, 32, 69, 110], is not sufficient because it oversimplifies Semantic Web data access by skipping the non-trivial step of obtaining the desired Semantic Web data maintained by independent owners on the Web.

In order to capture the Web aspect and support Web-scale Semantic Web data access, the current model of the Semantic Web should be enhanced to meet the following requirements:

- preserve the freedom of Web-based publishing, i.e., any one can publish any Semantic Web data on the Web as long as they provide appropriate Web interface
- model both the *content*, i.e., Semantic Web data that is represented in logical RDF graph, and the *context*, i.e., the Web that serializes the content and the agents that publish and consume the content
- track knowledge provenance, e.g. the locations and creators of Semantic Web data

II.B Harvesting the Semantic Web

Most existing works [48, 118, 151, 28, 25, 56, 125] obtain small amount of Semantic Web data from well-known sources such as (i) *online registries hosting manually submitted Semantic Web data*, e.g., DAML Ontology Library[1], SchemaWeb[2], Simile project RDF data collection[3], Protege ontology library[4] and RDF Schema Registry[5]; (ii) *popular ontologies*, e.g., RDFS, OWL, Dublin Core Element, and FOAF; (iii) *popular RDF dump of knowledge base or database*, e.g., DMOZ RDF dump[6] of Open Directory Project category database, and OpenCyc ontology[7] from CYC knowledge base [102]. We also observe that users generate Semantic Web data by (i) extracting data from conventional Web documents, e.g., SWETO dataset[8] [2], and TAP KB[9] [63]; and (ii) artificially synthesizing data, e.g. Lehigh University Benchmark (LUBM) [68].

The Semantic Web data observed from the above sources, however, is only a small part of the real world Semantic Web. The addresses of most Semantic Web data on the Web remain unknown to the information consumers. Therefore, effective automated methods are needed to harvest more Semantic Web data on the Web. To this end, Web crawling, which is the primary method in Web harvesting, has been experimented in Semantic Web literature: (i) Eberhart's RDF crawler [48] reported 1,479 Semantic Web documents in 2002. (ii) OntoKhoj [127] reported 418 Semantic Web ontologies[10] in 2003. (iii) DAML crawler [34] reported 21,021 DAML[11] pages in 2004. (iv) A *scutter* [44] reported 1.5 million of FOAF documents in 2005. Unfortunately, the datasets found by the above *Semantic Web crawlers* (aka. RDF crawlers [5]) are either in small amount or lack of diversity (e.g. being dominated by FOAF documents).

In order to build a global catalog of the Semantic Web, an automated Semantic Web crawler is needed with the following requirements:

- harvest significant amount of Semantic Web data from the Web effectively.
- preserve the diversity of the harvest result and reduce the bias introduced by the overwhelming usage of FOAF and RSS.

[1] `http://www.daml.org/ontologies/`
[2] `http://www.schemaweb.info/`
[3] `http://simile.mit.edu/repository/datasets/index.html`
[4] `http://protege.stanford.edu/plugins/owl/owl-library/`
[5] `http://139.91.183.30:9090/RDF/Examples.html`
[6] `http://rdf.dmoz.org/rdf/`
[7] `http://www.cyc.com/2004/06/04/cyc`
[8] `http://lsdis.cs.uga.edu/projects/semdis/sweto/`
[9] `http://tap.stanford.edu/tap/tapkb.html`
[10] The ontologies may be distributed since they are merged from URI references sharing the same namespace.
[11] The acronym of the DARPA Agent Markup Language, see `http://www.daml.org/`.

II.C Measuring the Semantic Web

Since the Semantic Web is on the Web, we can certainly reuse benchmarks for measuring the Web such as the size and age of a document from Web characterization literatures [131, 100]. However, these benchmarks do not show insights on the semantic aspect of the Semantic Web, such as the distribution of instance space.

Existing works on measuring Semantic Web data usually measure the quality of a specific class of data; moreover, the datasets used in experiments are often too trivial to draw convincing conclusions.

- **Measuring the quality of Semantic Web ontology** is a complicate issue and has attracted significant amount of works in literature. According to recent surveys [78, 55], most ontology evaluation studies employ content analysis with various focuses, e.g., building comprehensive evaluation framework [103], qualifying concept-consistency [157, 126], quantifying graph structure of class and property taxonomy hierarchy [107, 149, 1, 162], and quantifying the structure and the instance space of a given ontology [150]. However, their experiments only evaluate trivial number of real world ontologies. For example, only eight anonymous ontologies have been evaluated in [149], and the experiments neither explain the intuition nor justify the validity of the proposed metrics. Only three ontologies and corresponding instance data have been evaluated in [150], and the instances are trivial because they are generated by the ontology creators themselves using information extraction.

- **Characterizing the universal RDF graph** is approached by [56] which analyzes the structure of an RDF graph. However, the sample dataset is merged from only 196 ontologies obtained from DAML Ontology Library, so neither the size nor the diversity of the sample is significant enough to predict the global structural properties of the Semantic Web.

- **Characterizing the social network encoded by FOAF documents** emerges recently [89, 60, 111, 44] and it is approached by statistical measurements on vocabulary usage and network structure of FOAF documents obtained by RDF crawling. The evaluation dataset is quite large; however, it is dominated and biased by several websites, e.g. *www.livejournal.com*.

In order to measure the Semantic Web, we need a significant sample dataset of the real world Semantic Web to reveal interesting and credible (global) properties of the Semantic Web. The properties should be carefully chosen to provide insights on the current deployment status of the Semantic Web and to guide Semantic Web based applications.

II.D Facilitating Web-scale Semantic Web Data Access

In Web-scale Semantic Web data access, independent owners publish Semantic Web data across the Web, but most information consumers are lack of knowledge on the location of the published data. Moreover, information consumers have various data access requirements and treat the Semantic Web as a part of the Web or a unified database or knowledge base. There are quite a few works in literature related to Web-scale Semantic Web data access:

- **Semantic Annotation Systems**[12] help users to annotate and locate Web documents using semantic markups based on predefined ontologies. They provide an alternative search approach to full-text search: users may use structural query to locate semantic annotation entries and then navigate to the associated Web documents [33]. Early semantic annotation systems, e.g. SHOE [105], Ontobroker [36], WebKB [108], and Annotea [90], mainly rely on manually submitted semantic descriptions. The practical problems in manual approaches [49] motivate recent progress on semi-automatic annotation mechanisms, such as AeroDAML [96], OWLIR [141], CREAM [73, 75], SCORE [144], SemTag and Seeker [39], and KIM [132]. However, none of these semantic annotation systems is sufficient since they do not parse or annotate the semantic web content and structure of Semantic Web documents.

- **Ontology Library Systems**[13] manage a collection of ontologies for ontology evolution and reuse. We investigate two important classes of ontology library systems in literature: (i) *General ontology management systems* provide comprehensive ontology management services such as storage, modification, versioning, reasoning, search and browse. Example systems are Ontolingua [61, 50], SHOE [105], and KAON [106]. These systems, however, are incompatible with the Web aspect because their centralized designs contradict the decentralized ontology publishing requirements. (ii) *Semantic Web ontology systems* focus on browsing and searching existing Semantic Web ontologies. DAML Ontology Library and SchemaWeb provide browse and search interface for accessing a few hundreds of manual submitted Semantic Web ontologies; however, they depend on manual ontology submission. OntoKhoj [127] crawls the Web for additional Semantic Web ontologies but its crawling performance (418 ontologies were found out of 2,018,412 Web documents) is neither efficient nor effective. OntoSearch [164]

[12] Handschuh and Staab [74] cover both manual and semi-automatic approaches and several application areas of semantic annotation. Reeve and Han [135] conduct a comparative survey on semi-automatic semantic annotation approaches.

[13] Ding and Fensel [45] survey general ontology library systems in 2001. However, the report is a little bit out of date.

searches Semantic Web ontologies by querying Google APIs[14] using the concatenation of user's query string and a *filetype:rdfs* constraint; however, the approach is incomplete because Semantic Web ontologies may use *owl* or *rdf* as file extension or just have no file extension (e.g. RDFS ontology). These existing systems are insufficient for Web-scale data access because (i) only a small portion of online Semantic Web ontologies can be accessed via them, (ii) they offer limited data access features, and (iii) their development (except SchemaWeb) stopped at the prototype stage.

- **Semantic Web Databanks** host Semantic Web data using certain semantic storage system and provide Web interface for data access. For example, *Semantic Web Search*[15] lets users search instances of well-known RDF classes, such as *foaf:Person* and *rss:Item*, and it is backed by an RDF database and an RDF crawler. W3C's Ontaria [84] lets users search and browse the several hundreds of RDF graphs harvested from the Web. These systems are based on semantic storage systems and store all indexed RDF graph in whole, so they have the same difficulties as the semantic storage systems in enabling Web-scale Semantic Web data access.

- **Semantic Web Browsers** are the client side tools in Semantic Web data access. None of them can solve the accessibility issues alone, but they do help us to capture users' behaviors in accessing Semantic Web data on the Web. Quan and Karger [134] classify two types of browsers: (i) the Semantic *Web browsers* which use Semantic Web based annotation to facilitate Web browsing, such as Magpie [47] and Amaya[16], and (ii) the *Semantic Web browsers* that actually browse Semantic Web data, such as Hyperdaml[17] that converts Semantic Web documents to HTML documents by adding hyperlinks, FOAF explorer[18] and FOAFnaut[19] which translate FOAF documents into human friendly HTML documents and SVG (Scalable Vector Graphics) pictures respectively, Haystack [134] that displays aggregated Semantic Web data from the Web, Swoop [91] that helps users to view, edit and debug online SWDs.

Based on these existing works, we summarize three required but not yet fulfilled issues in Web-scale Semantic Web data access:

First, a new conceptual model is needed to model the unique behaviors in Web-scale Semantic Web data access. We have observed three types of behaviors in Web-scale Semantic Web data access: (i) **search**

[14] http://www.google.com/apis/
[15] http://www.semanticwebsearch.com/
[16] http://www.w3.org/Amaya/
[17] http://www.daml.org/2001/04/hyperdaml/
[18] http://xml.mfd-consult.dk/foaf/explorer/
[19] http://www.foafnaut.org/

[127, 164], which uses natural language based queries (especially keyword queries) to retrieve Semantic Web data, e.g. searching for all ontologies by keyword *person*; (ii) **query** [71], which composes queries in Semantic Web query languages (most existing query languages support inference, e.g. [82] and TRIPLE [146], but recent standardization work promotes SPARQL [133] which removes inference support for simplicity); and (iii) **navigation** (aka. browsing) [114], which is a special type of *query* with restricted patterns, e.g. enumerating all RDF nodes whose *rdf:type* is *foaf:Person*. When accessing the Semantic Web, human users prefer search and navigation to avoid the use of Semantic Web languages, but machine agents, prefer query and navigation to utilize Semantic Web languages. Recent works [64, 138, 163] further investigate hybrid methods for combining search and query, e.g. supporting the mixture of keyword query and structure query. However, existing works are lack of explicit model of these behaviors and only provide partial support.

Second, an effective search engine is needed to support Web-scale Semantic Web data access. The real world Semantic Web data, including ontologies and instance data, is in huge amount and is stored on the Web; therefore, maintaining a copy of all Semantic Web data in a single semantic storage system would violate the Web aspect as well as be computationally impossible. Therefore, a Semantic Web search engine is needed because it only maintains the compact metadata (including content description and knowledge provenance) of the distributed Semantic Web data and thus helps users to obtain Web addresses of Semantic Web data. None of existing works (including semantic annotation systems, ontology library systems, and Semantic Web databanks) except Swoogle [43] has built a substantial Semantic Web search engine that actively indexes a significant amount of Semantic Web data on the Web.

Third, ranking is needed to bring order to the Semantic Web. Most works in ranking Semantic Web ontologies employ content analysis based approach [107, 149, 1, 162, 150]; however, content quality is not the only factor in ordering ontologies because popularity is also important. Google's PageRank [123] quantifies a Web document's *popularity* from the Web's graph structure, and it turns out to be enormously useful in practice. Moreover, PageRank cannot be directly used in ranking Semantic Web data due to the uniqueness of Web-scale Semantic Web data access: (i) The navigation network is not merely a web of document because the logic RDF graph in each Semantic Web document comprises additional navigation network. (ii) Semantic Web surfing is rational because the semantics of Semantic Web languages may add more navigation preferences to random surfing behaviors. Existing link analysis based Semantic Web rankings [127, 40] are limited by their unexplainable heuristic weighting-scheme; therefore, we propose an explainable link based ranking based on the characteristic behaviors in Semantic Web data access.

II.E Provenance of Semantic Web Data

Provenance has been studied in digital library [46], database systems (e.g. data provenance [22] and view maintenance [29]), and artificial intelligence (e.g. knowledge provenance [31, 54] and proof tracing [30]). It refers to "*the place of origin*" according to WordNet 2.1[20]. In the Semantic Web, provenance has finer meanings, e.g., the Semantic Web document that contains a certain RDF graph, the creator of a Web document, and the Semantic Web document that defines a term. Besides annotating the meta-descriptions and relations of Semantic Web data, provenance data can be used in trust inference (e.g. trust propagation [140, 42]) and grouping RDF triples [147].

An appropriate level of granularity is critical in tracking the provenance of an RDF graph because we do not want to miss any Semantic Web documents that support the present RDF graph in part, or mistakenly include irrelevant Semantic Web documents, for example, the statement "*there exists a person whose last name is 'ding'*" should not be counted as supporting the statement "*there exists a person who has last name 'ding' and first name 'li'*".

The granularity issue has been investigated in many existing works such as (i) RDF graph matching [24], (ii) canonical representation of RDF graphs [25], (iii) RDF graph versioning [11], (iv) concise bounded description (CBD) [147], and (v) ontology partition [148, 59]. The first three applications focus on partitioning an RDF graph into small pieces without losing meaning and generating a canonical version. They are the closest to our work but have different application domains. The fourth one, CBD, is an ad hoc granularity that groups all triples related to the definition of a resource; however, this granularity can still be refined. The last one, ontology partition, need to partition large ontologies into independent (or less dependent) components to refine inter-ontology importing relations [155] or to study the *e-connections* [99, 98] between components. Ontology partition can be approached by structural method [148] and semantics based method [59]; however, a partitioned component usually consists of several semantically dependent classes and properties and can be further decomposed.

Our work can be thought of as a systematic enhancement to existing works on RDF graph decomposition [25, 11], and it additionally demonstrates the important application of lossless RDF graph decomposition in Semantic Web knowledge aggregation.

[20] source: wordnet 2.1, http://wordnet.princeton.edu/perl/webwn (2006-03-31)

Chapter III

MODELING THE SEMANTIC WEB

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. "

– Tim Berners-Lee, James Hendler and Ora Lassila [13]

The Semantic Web is widely recognized as a knowledge representation infrastructure [37, 13] where knowledge is represented in RDF graph using Semantic Web languages and is stored on the Web. Most existing works model the Semantic Web as one universal RDF graph. This view is good for studying the semantic aspect but insufficient for capturing the Web aspect. In order to grasp the Web aspect of the Semantic Web, we need to enhance the current model by acknowledging the fact that Semantic Web data is distributed on the Web and building appropriate ontologies for explicit meta-description of the Semantic Web.

III.A The Semantic Web on the Web

The Semantic Web is actually the *Semantic Web on the Web* because it is designed to support knowledge sharing across Web based applications [13]. According to the Web aspect, online Semantic Web data is addressed by URLs and accessed in the form of Semantic Web documents. Therefore, the Semantic Web is essentially a huge collection of static or dynamic Semantic Web documents distributed throughout the Web. The rest of this section uses a simple example to identify the key concepts and relations in Web-scale Semantic Web data access.

III.A.1 A Semantic Web Data Access Example

Semantic Web data access is a collaborative process where independent agents asynchronously create, publish and consume Semantic Web data on the Web. The following example demonstrates some frequently observed activities in Semantic Web data access.

Publishing Semantic Web data. A human user *Bob* published his knowledge about a person whose name is "Li Ding" in the following four steps.

1. determining the knowledge to be published (as shown below)

   There exists a *foaf:Person* with *foaf:name* "Li Ding" and *foaf:email* "mailto:dingli1@umbc.edu", and the person is the same as the other resource referred by the URI reference `http://cs.umbc.edu/~dingli1/foaf.rdf#dingli`".

2. encoding the knowledge in an RDF graph using Semantic Web languages such as OWL and Semantic Web vocabulary such as FOAF (Figure. III.1)

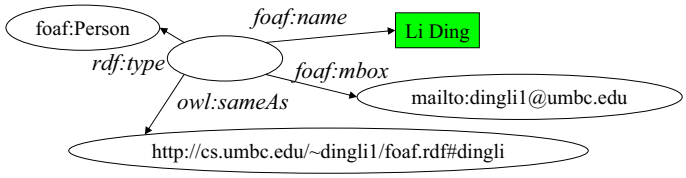


Figure III.1: An RDF graph about "Li Ding"

3. serializing the RDF graph in a Semantic Web document using RDF/XML grammar [8] (Figure III.2)

```
 1: <?xml version="1.0" encoding="utf-8"?>
 2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 3:          xmlns:owl="http://www.w3.org/2002/07/owl#"
 4:          xmlns:foaf="http://xmlns.com/foaf/0.1/" >
 5: <foaf:Person>
 6:   <foaf:name>Li Ding</foaf:name>
 7:   <foaf:mbox rdf:resource="mailto:dingli1@umbc.edu"/>
 8:   <owl:sameAs rdf:resource="http://cs.umbc.edu/~dingli1/foaf.rdf#dingli"/>
 9: </foaf:Person>
10: </rdf:RDF>
```

Figure III.2: The RDF/XML Semantic Web document about "Li Ding"

4. publishing the Semantic Web document on the Web with URL `http://ebiquity.umbc.edu/get/a/resource/134.rdf`.

Consuming Semantic Web data. Some time later, a software agent *iRobot* found and downloaded the document published by *Bob*. *iRobot* optionally checked its trusts in *Bob*, and concluded that the RDF graph in the document is trustworthy. This step is useful because not all Semantic Web data is necessarily true. Once *iRobot* had believed the RDF graph, it then added the RDF graph to its knowledge base. In order to reason about the unique identifiers of the instance of *foaf:Person*, *iRobot* further pursued term definitions from FOAF ontology `http://xmlns.org/foaf/0.1/index.rdf`, and then found that *foaf:mbox* can be used to uniquely identify an instance of *foaf:person* because its *rdf:type* is *owl:InverseFunctionalProperty*.

III.A.2 Identifying Concepts and Relations

From the example in Section III.A.1 , the RDF graph world is not the only world involved in Web-scale Semantic Web data access because the context of Semantic Web data (i.e., the Web and the agent world) is also an important part. Therefore, we summarize three worlds as the following:

- **The RDF graph world** is the logical layer for sharing Semantic Web data. *RDF resource* enables mappings between the concepts in our mind and the terms in RDF graph world; moreover, the ontology constructs provided by Semantic Web languages offer rich semantics for building Semantic Web vocabulary. *RDF graph* enables assertions about RDF resources. Therefore, agents may represent their beliefs about the world using RDF graph.

- **The Web** is the physical layer for publishing and consuming Semantic Web data. In the Web, Semantic Web document is the transfer packet of Semantic Web data, and its meaning has two senses: serializing an RDF graph and being a Web document addressed by URL.

- **The agent world** hosts the users of Semantic Web data. In this world, agents create, publish, and consume Semantic Web data. Moreover, agents may hold belief states on Semantic Web data as well as on other agents, and they may selectively acquire Semantic Web data from the Web.

These three worlds are not necessarily the only worlds involved in Web-scale Semantic Web data access because other works [94, 66] have shown more complex context of Semantic Web data, e.g., social context for tasks modeling [80] and temporal context for ontology versioning [93]. In this dissertation, we focus on the Web and the agent World because they are two fundamental components for any further characterization of the context. In Figure III.3, we identify five important concepts in the three worlds, namely *agent*, *person*, *RDF resource*, *RDF graph*, and *Semantic Web document*. We also identify three important categories of

relations, namely (i) *provenance*, which captures the knowledge provenance relations within and across these worlds, (ii) *belief*, which captures agents' belief states about Semantic Web data and other agents; and (iii) *subClassOf*, which captures class hierarchy and supports property inheritance.

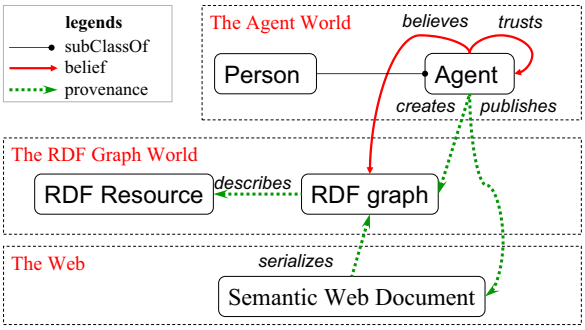


Figure III.3: Important concepts and relations in the Semantic Web and its context

The instances of the five concepts can be found in the example in Section III.A.1:

- *Bob* is an instance of **person** in the agent world.
- *iRobot* is an instance of **agent** in the agent world.
- *foaf:mbox* is an **RDF resource** in the RDF graph world.
- An **RDF graph** by itself is an important concept in the RDF graph world.
- the URI `http://ebiquity.umbc.edu/get/a/resource/134.rdf` references an instance of **Semantic Web Document** in the Web.

The three types of relations are instantiated in the following scenarios in Section III.A.1

- (Provenance) A person *Bob* **creates** an RDF graph.
- (Provenance) The RDF graph **describes** an RDF resource *foaf:Person*.
- (Provenance) A Semantic Web document **serializes** the RDF graph, and *Bob* **publishes** the document on the Web with URL `http://ebiquity.umbc.edu/get/a/resource/134.rdf`.
- (Belief) The agent *iRobot* accepts the RDF graph created by *Bob* because it **believes** the graph.
- (Belief) The agent *iRobot* **believes** *Bob*'s knowledge because it **trusts** *Bob*. Here, *trust* estimates the overall trustworthiness of knowledge created by the agent.
- (subClassOf) The concept *person* is indeed a **subClassOf** the concept *Agent*.

III.B The Web of Belief Ontology

We enhance the current model of the Semantic Web to acknowledge the Web aspect, and the new model is supported by a Semantic Web ontology for explicit knowledge representation. The ontology is named after one of W. V. Quine's book "*Web of Belief*" because the Semantic Web is indeed a *web of belief* where agents publish and consume the believed knowledge encoded in RDF graph on the Web and an agent seldom blindly believes in all published knowledge.

The *Web Of Belief* (WOB) ontology provides a series of ontology constructs for characterizing the concepts and relations identified in the previous section. It also enables explicit meta-description about the Semantic Web on the Web and thus helps us to address the accessibility and data quality issues in Web-scale Semantic Web data access. For the *accessibility* issue, various provenance relations are explicitly represented to enrich navigational paths and facilitate accessing Semantic Web data on the Web. For the *data quality* issue, knowledge provenance and agents' belief states are explicitly represented to facilitate aggregating Semantic Web data from sources with varying data quality.

The WOB ontology is designed to meet the three requirements mentioned in Chapter II, i.e. preserving Web publishing mechanism, modeling the content and context of Semantic Web data, and tracking knowledge provenance. Moreover, the implementation of the WOB ontology is guided by the following principles:

- **principle 1 – being a core ontology**. The WOB ontology only defines the most generic and important concepts and relations, and other specific concepts can be defined in extension ontologies.

- **principle 2 – being computable**. The WOB ontology is mainly used by machines; therefore, it must be encoded using Semantic Web ontology languages; it should promote usage of URIs in knowledge representation because URIs are better than textual annotations for machine process; and it should better use ontology constructs no more than OWL DL (or OWL Lite) [35].

- **principle 3 – being reusable**. The WOB ontology should not be developed from scratch; instead, it should reuse as many as possible definitions from existing popular ontologies. Moreover, it is also useful to create mapping between WOB concepts and similar concepts from ontologies to enrich the instance space of the Semantic Web.

III.B.1 Semantic Web document

Definition 1 (Semantic Web document (SWD)) *wob:SemanticWebDocument is a class of Web documents serializing one or several RDF graphs. The URL of a Semantic Web document has three senses: (i) the address of the document on the Web, (ii) the unique identifier of the document in RDF graph world, and (iii) the unique identifier of the RDF graph serialized in the document.*

A Semantic Web document, according this definition, is an atomic Semantic Web data transfer packet on the Web regardless of how the data is actually stored behind the Web server. Even though many Semantic Web data are directly stored in static files, there are still many Semantic Web documents dynamically generated from, for example, database query results and agent status reports.

Intuitively, the concept *Semantic Web document* replicates the idea of hyperlink in the Web: each instance of *wob:SemanticWebDocument* in RDF graph forms an explicit hyperlink to a Semantic Web document while each URL in Web document forms a hyperlink to a Web document. Since many URIs in an RDF graph are in fact linking to Web documents, annotating the rest URIs that link to Semantic Web documents will greatly improve Semantic Web surfing experiences such as Semantic Web crawling.

In order to annotate parse instructions for a Semantic Web document, we further define two properties:

- **wob:hasGrammar**. It indicates the syntactic RDF grammar used to encode a Semantic Web document. Currently, the WOB ontology enumerates three possible values for this property according to W3C's recommendations, namely RDF/XML [8], Notation 3 (N3) [9], and N-Triples (NT) [58]. This property helps agents to avoid the computational overhead for guessing the RDF grammar; in fact, similar practices have found in RDF Test [58] (e.g. *rdftest:RDF-XML-Document*) and OWL test [27].

- **wob:isEmbedded**. It indicates whether the Semantic Web data is embedded in a Web document. While pure Semantic Web documents are completely written in Semantic Web languages, some conventional Web document may embed some Semantic Web data [124], e.g., a span of text serializing a small RDF graph in an HTML document. Examples of embedded Semantic Web document are HTML documents containing Creative Commons license metadata[1] and PDF documents containing XMP metadata[2].

Although many relevant definitions have been found in popular ontologies, none has exactly the same semantics as *wob:SemanticWebDocument*.

[1] `http://creativecommons.org/technology/metadata/`
[2] `http://www.adobe.com/products/xmp/main.html`

- `http://www.w3.org/2002/07/owl#Ontology` refers to Semantic Web documents that define Semantic Web ontologies. Hence, it is a sub-class of *wob:SemanticWebDocument*.

- `http://xmlns.com/foaf/0.1/Document` refers to a general class of documents ranging from Web documents to physical publications. Hence, *wob:SemanticWebDocument* is its sub-class. Meanwhile, this concept is defined as a sub-class of *wn:Document* in FOAF ontology.

- `http://inferenceweb.stanford.edu/2004/07/iw.owl#Source` refers to a general class of knowledge sources such as agents, teams, organizations, websites, publications, and Web documents. Hence, it is a super-class of *wob:SemanticWebDocument*.

- `http://xmlns.com/foaf/0.1/PersonalProfileDocument` refers to Semantic Web documents that describe personal profiles. Hence, it is a sub-class of *wob:SemanticWebDocument*.

- `http://www.w3.org/2000/10/rdf-tests/rdfcore/testSchema#RDF-XML-Document` refers to Semantic Web documents that use RDF/XML as RDF grammar; however, it is mainly used in RDF test and OWL test. Hence, it is a sub-class of *wob:SemanticWebDocument*. There are still many similar definitions using the same namespace, such as *inputDocument*, *conclusionDocument*, *premiseDocument*, *NT-Document*, *ConsistencyTest*, and PositiveEntailmentTest, or using other namespaces, such as `http://www.daml.org/services/owl-s/1.0/Grounding.owl` and `http://www.w3.org/2003/03/rdfqr-tests/query.rdfs`.

Figure III.4 summaries the properties (rectangles) of *wob:SemanticWebDocument* via *domain* arcs, and shows how it relates to other existing classes (round-corner rectangles) via *subClassOf* arcs. Those with grey background are new concepts defined in the WOB ontology.

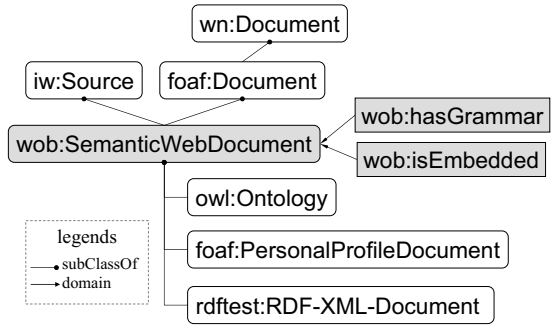


Figure III.4: Classes and properties related to *wob:SemanticWebDocument*

III.B.2 RDF Graph Reference

Definition 2 (RDF Graph Reference) wob:GraphReference *is a class of Semantic Web entities, each of which references an arbitrary RDF graph. A graph reference is not a replica of the referenced graph; indeed, it contains necessary information for reconstructing the referenced graph. Two instances of graph reference are equivalent if they are syntactically identical or their generated RDF graphs are semantically equivalent.*

RDF Graph reference plays an important role in addressing RDF graph in the Semantic Web. It is highlighted by two features: (i) it enables citation mechanisms on the Semantic Web such that users can cite an arbitrary sub-graph of any published RDF graph using an instance of *wob:GraphReference*; and (ii) it can be extended to represent required metadata for accessing any online Semantic Web data sources including RDF databases based Web services. Even though special tools may be needed by information consumers to process the data access instructions specified by the instances of *wob:GraphReference*, the representation of the *Graph Reference* concept does not extend the current RDF infrastructure.

In this dissertation, we define *wob:GraphReference* as the most general super-class for referencing RDF graph[3] , and then show how existing approaches can be integrated as its extensions without altering RDF.

- **Simple graph reference using document URL.** Many existing works such as RDF test simply reference an RDF graph by the Semantic Web document that serializes the graph. Therefore, the WOB ontology defines *wob:SemanticWebDocument* as a sub-class of *wob:GraphReference*. For example, the entire RDF graph serialized by a Semantic Web document shown in Figure III.5 can be directly referenced by an instance of *wob:SemanticWebDocument* whose URI is `http://www.cs.umbc.edu/~dingli1/foaf.rdf`.

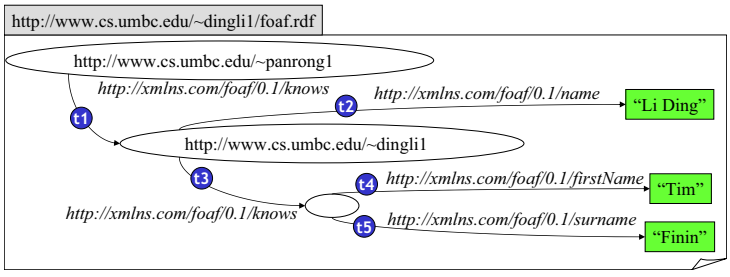


Figure III.5: An example Semantic Web document and its RDF graph

[3]We have also found related definitions of *graph*; however, they are not intended to reference RDF graph, e.g., `http://purl.org/puninj/2001/05/rgml-schema#Graph` for geometric graph, and `http://www.mygrid.org.uk/ontology#graph` or `http://www.cyc.com/2004/06/04/cyc#DirectedGraph` for generic graph concept.

- **Reified statement**. *rdfs:Statement* is defined in *RDF Schema* [19] for referencing RDF graph at triple level. Therefore, the WOB ontology defines it as a sub-class of *wob:GraphReference*. For example, the instance of *rdfs:Statement* in Figure III.6 references the triple *t2* in Figure III.5. However, this approach cannot reference the two triples (t4 and t5) connected by a blank node in Figure III.5 without losing the binding semantics of the blank node.

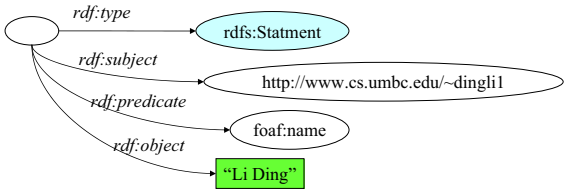


Figure III.6: Referencing RDF graph using reification

- **Named Graph**. *Named Graph* [26] lets publishers embed multiple RDF graphs, each of which groups a set of triples, in one Web document (see Figure III.7). `http://www.w3.org/2004/03/trix/rdfg-1/Graph` is defined to reference “*An RDF graph (with intentional semantics)*”; however, its instances cannot be serialized by any of the three recommended RDF grammars because the semantics of *Named Graph* is one degree higher than current RDF infrastructure. Moreover, the publishers determine the named graphs at creation time, and we still cannot freely reference the sub-graph of a named graph. Therefore, the WOB ontology does not support the *Named Graph* approach.

```
:G1 {  _:Monica ex:name "Monica Murphy" .
       _:Monica ex:email <mailto:monica@murphy.org> .
       :G1 pr:disallowedUsage pr:Marketing }

:G2 {  :G1 ex:author :Chris .
       :G1 ex:date "2003-09-03"xsd:date }
```

Figure III.7: An example Named Graph borrowed from [26]

- **Concise Bounded Description.** *Concise Bounded Description* (CBD) [147] lets users extract all triples that define a given URI in a given RDF graph. This approach only references a few sub-graphs of an RDF graph with a 2-tuple (the RDF graph, a URI), and it requires a special query engine for parsing and reconstructing the referenced RDF graph. Therefore, we define a class *wobGraph:GraphReferenceCbd* in WOB’s graph reference extension: its source graph is referenced by *wob:hasSourceGraph*; and the URI is stored in *wobGraph:hasCbdQuery*. The instance of *wobGraph:GraphReferenceCbd* in Figure III.8 references all triples except *t1* in the example RDF graph in Figure III.5.

- **SPARQL.** SPARQL [133] is the latest standard work lead by W3C. Among many RDF query lan-

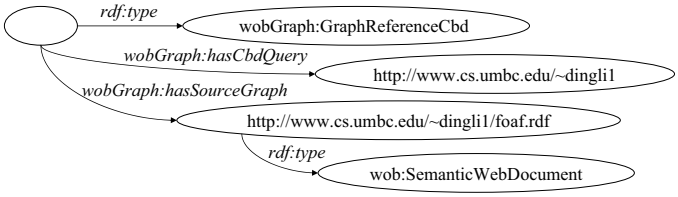


Figure III.8: Referencing RDF graph using Concise Bounded Description

guages in literature [71], such as RDQL [139] and RQL [92], SPARQL is especially useful in supporting referencing RDF graph. First, a SPARQL query using the *CONSTRUCT* operator and *FROM* clauses can reference an RDF graph. We define a class *wobGraph:GraphReferenceSparql* in WOB's graph reference extension: the source graphs are referenced by *wob:hasSourceDocument*; and the SPARQL query (except the FROM part) is stored in *wobGraph:hasSparqlQuery*. The instance of *wobGraph:GraphReferenceSparql* in Figure III.9 references the triple *t1* in Figure III.5.

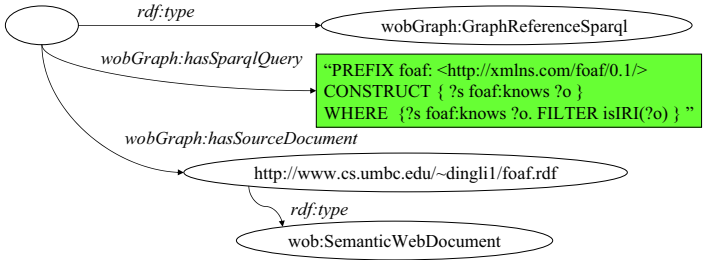


Figure III.9: Referencing RDF graph using SPARQL query

Second, the *WHERE* clause of a SPARQL query alone can be used to reference an arbitrary sub-graph of an RDF graph. We define a property *wobGraph:hasSparqlFilter* to store the WHERE clause (and the prefix declaration) and use it to extract the desired sub-graph from RDF graph reconstructed from an instance of *wob:GraphReference*. The instance of *wob:SemanticWebDocument* in Figure III.10 references two triples *t4* and *t5* in Figure III.5.

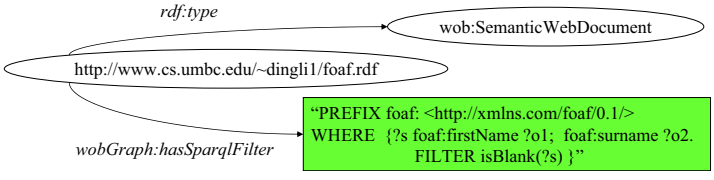


Figure III.10: Referencing sub-graph using SPARQL query

Usually, information consumers can download the source RDF graph and use a local query processor to extract the sub-graph according to SPARQL query; however, publishers may also process SPARQL query at server side to reduce the overhead introduced by transferring irrelevant part of a huge RDF graph on the Web. We may enhance the graph reference extension in the future by representing the query and the parameters for accessing RDF graph published by Web services, e.g. Joseki (`http://www.joseki.org`).

- **Union**. The WOB ontology also defines a class *wob:GraphReferenceUnion* to provide a simplified approach to reference multiple RDF graphs without SPARQL involvement. A property *wob:isUnionOf* is defined with range *rdf:List*, each item of which is of type *wob:GraphReference*. An instance of *wob:GraphReferenceUnion* references an RDF graph which is merged from the RDF graphs referenced by its member *Graph References*.

It is notable that RDF graph reference could be out-of-date due to the change of the referenced RDF graph. Therefore, we reuse the term *dc:created* to record the creation time of the graph reference. A graph reference is valid either (i) when this property is absence or (ii) when its creation time is later than the last modified time of the referenced RDF graph(s).

Figure III.11 summaries the related properties and classes of *wob:GraphReference*. It is notable that *wob:hasSourceDocument* is the *rdfs:subPropertyOf wob:hasSourceGraph* due to the dual semantics of Semantic Web document.

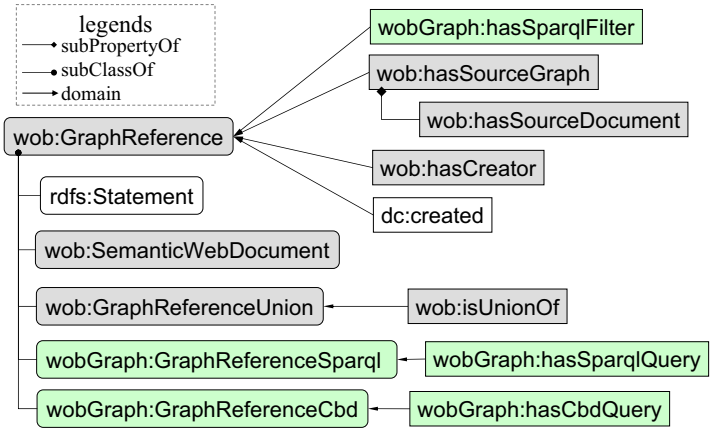


Figure III.11: Classes and properties related to *wob:GraphReference*

III.B.3 Semantic Web Term and Related Concepts

The WOB ontology reuses the definition of *rdfs:Resource*, which refers to the class of RDF resources used in RDF graphs. The *URI reference*[4] of resource plays an important role in connecting RDF graphs distributed on the Semantic Web. Besides being a unique symbol in Semantic Web, the URI of a resource is often used as a Web address for seeking the definition of the resource.

Semantic Web languages help users to represent knowledge in terms of classes and properties; hence, resources being populated, defined or referenced as classes or properties (see Section III.B.5) are in fact the backbone of Semantic Web vocabulary. The WOB ontology defines a sub-class of *rdfs:Resource* called *Semantic Web term* for classes and properties in Semantic Web vocabulary. Moreover, we can divide the Semantic Web documents into two sub-classes, namely *Semantic Web ontology* and *Semantic Web dataset*, by whether they contain definition of Semantic Web terms or not. Therefore, users may concentrate on the class and property definitions by studying only the Semantic Web ontologies.

Definition 3 (Semantic Web Term (SWT)) wob:SemanticWebTerm *refers to a special class of RDF resources, each of which has valid URI reference and has meta usage (e.g., being defined, referenced or populated as a class or a property) in at least one Semantic Web document.*

Definition 4 (Semantic Web Ontology (SWO)) wob:SemanticWebOntology *refers to a special class of Semantic Web documents that define at least one Semantic Web term.*

Definition 5 (Semantic Web Dataset (SWDS)) wob:SemanticWebDataset *refers to a special class of Semantic Web documents other than SWOs.*

While Semantic Web terms have been physically grouped by Semantic Web ontologies, they are semantically grouped by namespaces. Based on the observation that terms in the same namespace (e.g., FOAF namespace) had been defined in multiple Semantic Web ontologies, a namespace may not be exclusively owned by one Semantic Web ontology. In order to study the role of namespace, the WOB ontology defines a sub-class of *rdfs:Resource* called Semantic Web namespace.

Definition 6 (Semantic Web Namespace) wob:SemanticWebNamespace *refers to a special class of named RDF resources, each of which has been used as the namespace of some RDF resources.*

[4]URI reference is used interchangeably with URI in this dissertation.

III.B.4 Agent and Person

Agent is a widely used concept. Psychology, social science, and management science usually treat agent as *human agent* in the context of social community. Computer science, especially artificial intelligence, study *software agent* [88, 145] and its applications in various fields [154, 87, 70].

In the Semantic Web, *agent* usually refers to the creators, publishers and consumers of Semantic Web data. To describe the agent world, the WOB ontology covers only two important concepts: (i) the general concept *agent*, i.e. actionable entities, and (ii) a specific concept *person* which is being widely used. Other specific concepts in the agent world can be added in WOB's extension ontologies.

Our observation shows both *agent* and *person* have been defined in many Semantic Web ontologies. The two terms have been used as the exact local name of 232 and 647 Semantic Web terms respectively, and we have observed more (over 10,000) related SWTs when we use substring match[5]. Among these terms, *foaf:Agent* and *cc:Agent* are the most used. Although *foaf:Agent* is less populated as class than *cc:Agent*, it has significantly richer ontological definitions[6]. Further investigation on their usages shows that *cc:Agent* usually refers to a person while *foaf:Agent* turns out to be a more generic concept. In practice, the WOB ontology chooses *foaf:Agent* for the *agent* concept, and *foaf:Person* for the *person* concept.

III.B.5 Provenance

The WOB ontology is highlighted by its vocabulary in representing provenance in the Semantic Web. It defines a property *wob:provenance* as the most general *provenance relation*. We are particularly interested in provenance relations linking entities in the Semantic Web because they form the navigational paths in the Semantic Web. Figure III.12 depicts five types of specific provenance relations.

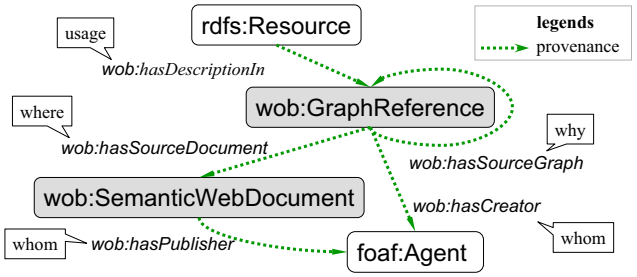


Figure III.12: Provenance relations defined in the WOB ontology

[5]The statistics are based on our harvested dataset as described in Chapter IV.
[6]Swoogle has found that the former is defined by 27 distinctive triples while the latter is defined by only 3 triples.

- **wob:hasDescriptionIn** (usage-provenance): The present resource is used in the source RDF graph.
- **wob:hasSourceGraph**(why-provenance): The present RDF graph can be derived from the source RDF graph in whole or in part. It is a sub-property of *dc:source*. It links two instances of *wob:GraphReference*.
- **wob:hasSourceDocument**(where-provenance): The present RDF graph may be derived from the RDF graph serialized by the source Semantic Web document in whole or in part. It is a sub-property of *wob:hasSourceGraph* and its *rdfs:range* is *wob:SemanticWebDocument*.
- **wob:hasCreator** (whom-provenance): the reference RDF graph is created by the source agent. *dc:creator* is defined as "An entity primarily responsible for making the content of the resource"; however, the "content of the resource" is not very clear. Hence, the WOB ontology introduces this new concept and restricts its *rdfs:domain* to *wob:GraphReference* and its *rdfs:range* to *foaf:Agent*.
- **wob:hasPublisher** (whom-provenance): the present Semantic Web document is published on the Web by the source agent. *dc:publisher* is defined as "An entity responsible for making the resource available"; however, it does not specify where and in what form the resource is available. Since agents share Semantic Web data using Semantic Web documents, the WOB ontology introduces this new concept and restricts its *rdfs:domain* to *wob:SemanticWebDocument* and its *rdfs:range* to *foaf:Agent*.

Although popular ontologies such as Dublin Core have defined most of the provenance relations, their definitions are not specific enough for meta-description of the Semantic Web. Therefore, the WOB ontology defines specific terms using OWL to facilitate machine process. Figure III.13 depicts how WOB concepts associate with existing terms.

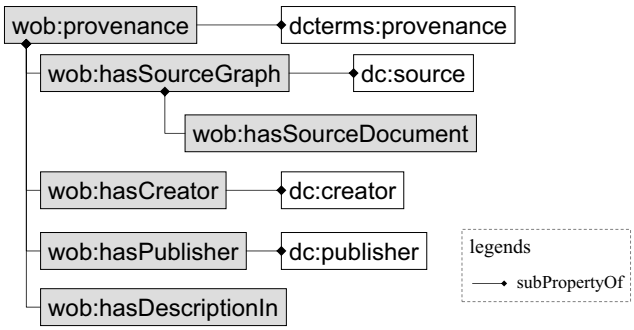


Figure III.13: Basic provenance relations in the WOB ontology

Meta-usages of Semantic Web Term

The semantics of an SWT depends on its usage in residential RDF graph. In order to study the meta-usage of SWTs, we define a property *wob:hasMetaDescriptionIn* and its six sub-properties as the following:

- *wob:hasClassDefinitionIn*: The present SWT is defined as a class in the target RDF graph. A resource X is defined as a class if there exists a triple like *(X, rdf:type, C)* where C is *rdfs:subClassOf rdfs:Class*. For example, *foaf:Person* is defined as a class according to triple *t3* in Figure III.14.

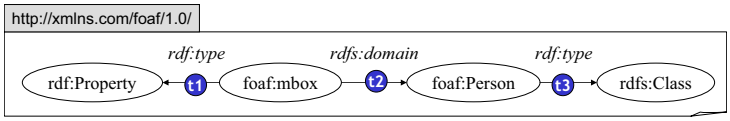


Figure III.14: An example RDF graph for illustrating meta-usage of SWT

- *wob:hasPropertyDefinitionIn*: The present SWT is defined as a property in the target RDF graph. A resource X is defined as a property if there exists a triple *(X, rdf:type, P)* where P is *rdfs:subClassOf rdf:Property*. For example, *foaf:mbox* is defined as a property by *t1* in Figure III.14.

- *wob:hasClassInstanceIn*: The present SWT is instantiated (i.e. populated) as a class in the target RDF graph. A resource X is populated as a class if there exists a triple *(_a , rdf:type , X)* where $\_a$ can be any resource. For example, $\_a$ is a class-instance of *rdfs:Class* according to triple *t3* in Figure III.14.

- *wob:hasPropertyInstanceIn*: The present SWT is instantiated as a property in the target RDF graph. A resource X is populated as a property if there exists a triple *(_a , X , _b)* where $\_a$ and $\_b$ can be any resource (or literal). For example, triple *t3* has an instantiation of the property *rdf:type* in Figure III.14.

- *wob:hasClassReferenceIn*: The present SWT is referenced as a class in the target RDF graph. A resource X is referenced as a class in an RDF graph if X is of type *rdfs:Class* according to the vocabulary of Semantic Web languages without the involvement of *rdf:type*. For example, *foaf:Person* is referenced as a class by triple *t2* in Figure III.14.

- *wob:hasPropertyReferenceIn*: The present SWT is referenced as a property in the target RDF graph. A resource X is referenced as a property in an RDF graph if X is of type *rdf:Property* according to the vocabulary of Semantic Web languages without the involvement *rdf:type*. For example, *foaf:mbox* is referenced as a property by triple *t2* in Figure III.14.

While *wob:hasClassInstanceIn* and *wob:hasPropertyInstanceIn* help users to capture the population of Semantic Web terms, the other four sub-types of meta-usage help users to capture the definition of an SWT distributed in multiple Semantic Web documents. For example, the term *rdfs:subClassOf* is defined as a property by the popular RDFS ontology; however, it is also defined as an *rdfs:Class* by a Semantic Web document `http://ilrt.org/discovery/2001/09/rdf-schema-tests/rdf-schema.rdfs`. Therefore, we should not completely rely on either the *Web addressing* semantics of a Semantic Web term's URI or the semantics of *owl:imports* to find the complete definition of a Semantic Web term; instead, we need a global catalog and effective tools to enumerate the occurrence of a term's URI and thus aggregate corresponding definitions.

III.B.6 Assertion, Belief and Trust

In addition to explicit representation of provenance relations, the WOB ontology also supports explicit representation of agent's belief states. *wob:AgentAssertion* is defined to capture an agent's belief states that are usually N-ary relations $(N > 2)$. It has two sub-classes: *wob:BeliefAssertion*, which is about an agent's belief state on a piece of Semantic Web data, and *wob:TrustAssertion*, which is about an agent's belief state on another agent's knowledge. Figure III.15 depicts the related classes and properties in the WOB ontology. The WOB ontology also defines three properties: (i) *wob:hasOwnerAgent* which links to the owner of the belief states, i.e., the agent who has made the assertion, (ii) *wob:hasTargetGraph* which links to the RDF graph asserted by the agent, and (iii) *wob:hasTargetAgent* which links to another agent asserted by the agent.

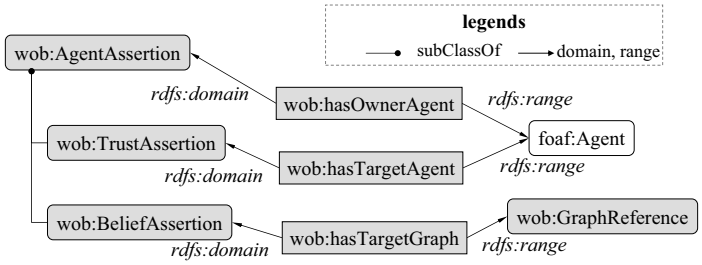


Figure III.15: Assertion, belief and trust in the WOB ontology

The WOB ontology only defines three most generic classes and three properties that connect *foaf:Agent* and *wob:GraphReference*, and it leaves the definition of the actual semantics of the assertions (i.e., the terms for representing agents' opinions) to extension ontologies. Therefore, it grants users the freedom to create their own belief/trust extensions according their preferred belief/trust representation/computation scheme.

III.C Summary

In this chapter, we present the WOB ontology that meets all the requirements mentioned in Chapter II:

- The requirement on preserving the freedom of Web publishing is mainly acknowledged by the term *wob:GraphReference* and its sub-classes, which explicitly represent the Web address of Semantic Web data. For example, *wob:SemanticWebDocument* lets users directly access Semantic Web data on the Web via URL. Moreover, this concept is highly extensible such that it can be used to represent the required information for accessing Semantic Web data sources even if they use ad hoc Web based protocols.

- The requirement on modeling both the content and the context of Semantic Web data is supported: (i) *wob:GraphReference* and *wob:SemanticWebTerm* help users to reference the content of Semantic Web data; and (ii) *foaf:Agent* and *wob:SemanticWebDocument* help users to reference the context.

- The requirement on knowledge provenance representation is supported in two folds: (i) the concepts (e.g., SWD and SWT) in the WOB ontology are mainly used to reference the entities in the Semantic Web and its context, and (ii) the five basic types of *provenance relation* help users to track various aspects of knowledge provenance.

Moreover, the WOB ontology meets the three design principles as the following: (i) it only defines core concepts; (ii) most properties are defined as *owl:ObjectProperty* to encode machine-friendly operational semantics and the core ontology is classified as OWL DL; and (iii) it reuses terms from FOAF ontology, and an OWL FULL extension maps its terms to terms defined in other popular Semantic Web ontologies.

The WOB ontology is not *yet another ontology* because it has made fundamental contributions in enhancing the current model of Semantic Web with respect to the Web aspect. No existing ontology but the WOB ontology systematically captures the core concepts and relations for describing the Semantic Web on the Web. For example, Semantic Web languages (RDF, RDFS, OWL and DAML) only focus on the RDF graph world; Dublin Core ontologies are too general because they aim at describing a much larger world than the Semantic Web; FOAF ontology [20] and SWPortal Ontology [115] concentrate on the meta-description about social networks in the agent world and related concepts in the Web; Web of Trust ontology[7] focuses on assuring content integrity of Web documents; and KSL's Proof Markup Language (PML) Ontology [30] focuses on explicit representation of proof traces and knowledge provenance.

[7] `http://xmlns.com/wot/0.1/`

Chapter IV

HARVESTING THE SEMANTIC WEB

Semantic Web technologies are making more Semantic Web data available on the Web; however, it is hard to obtain a global catalog of the Semantic Web due to two major difficulties: (i) Semantic Web documents are hard to be found on the Web because they are overwhelmed by conventional Web documents; and (ii) it takes non-trivial computation to confirm if a Web document contains Semantic Web data. Most existing approaches presented in Chapter II are not effective in harvesting Semantic Web data on the Web:

- Manual submission based approaches, such as DAML ontology library and SchemaWeb, have collected only hundreds of Semantic Web documents.
- Brute-force approaches, such as validating all Web documents indexed by Google, are inefficient because they waste huge amount of computational resources in validating conventional Web documents.
- Meta-search based approaches [164] are limited because Web search engines seldom differentiate Semantic Web documents from conventional Web documents and usually ignore semantic markups.
- Conventional HTML crawling [48, 127] has similar efficiency problems because most hyperlinks in Web documents (or even Semantic Web documents) link to conventional Web documents.
- RDF crawler (aka. scutter [18] or Semantic Web crawler) [34, 5] is limited and biased because it is hard to obtain seeding URLs (i.e., the starting point of crawling) and link indicators (i.e. heuristic patterns for selecting hyperlinks linking to SWDs).

IV.A A Hybrid Semantic Web Harvesting Framework

In order to harvest as many as possible Semantic Web documents on the Web with minimum cost, we design a hybrid Semantic Web harvesting framework. Figure IV.1 illustrates how the framework integrates several harvesting methods and achieve effective harvesting. Manual submission of URLs is used to bootstrap the seeds for Google based meta-crawling and bounded HTML crawling. These two crawlers are used to automatically collect the seeding URLs for RDF crawling. The RDF crawler visits the seeding URLs periodically to maintain an up-to-date picture of the Semantic Web, and selectively harvests new seeding URLs for itself using syntactic and semantic parsing results. The harvested SWDs are then used as training data to inductively generate new seeds for Google based meta crawling and bounded HTML crawling.

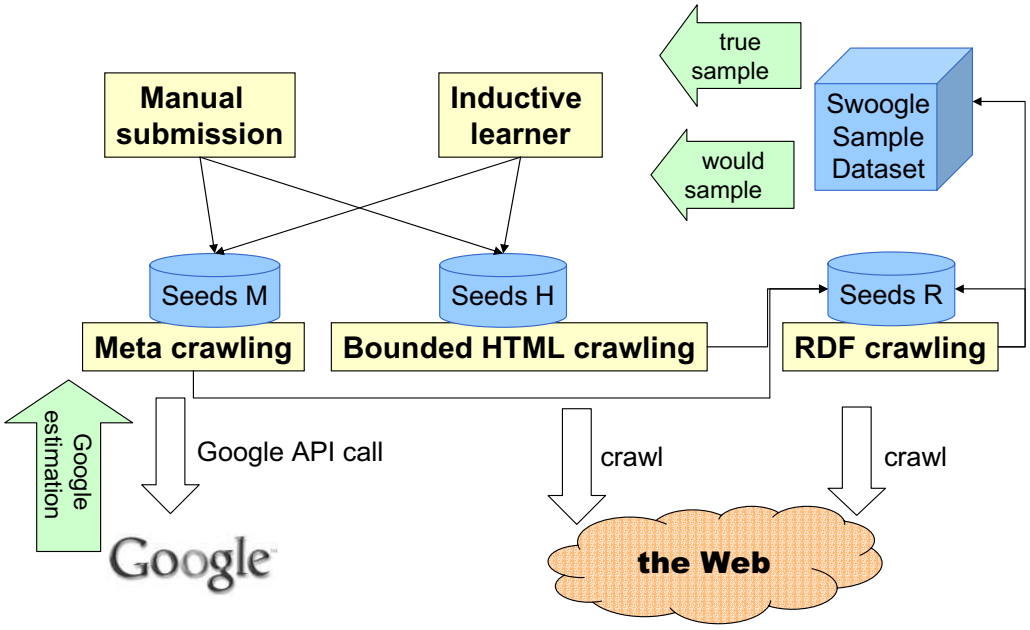


Figure IV.1: The architecture of the hybrid Semantic Web harvesting framework

IV.A.1 Automated Crawling

Besides manual submission, the hybrid Semantic Web harvesting framework employs three automated harvesting methods as the following:

- **Meta crawling.** *Meta crawling* [142, 143] queries conventional Web search engines and retrieves URLs of Web documents matching the query constrains. It is highlighted by directly harvesting URLs

without crawling the Web. Although meta-crawling retrieves both Semantic Web documents and Web documents, we still can construct some "good" queries whose results contain high percentage of SWDs. For example, most URLs returned by Google query *rdf filetype:rdf* are linking to SWDs. This feature leverages the limitation of RDF crawling by automatically collecting seeding URLs for RDF crawling. In order to obtain "good" queries for meta-crawling, manual submissions may not be enough. Therefore, we employ the inductive learner to automatically generate "good" queries from the sample dataset.

- **bounded HTML crawling.** We refer *HTML crawling* to *conventional Web crawling*, which is useful in exhausting a cluster of connected Semantic Web documents. Although it is in general inefficient because Semantic Web data is sparsely distributed on the Web, we do observe some websites having dense Semantic Web data distribution. For example, `http://iw.standford.edu/proofs` hosts many PML documents[1]. In order to control the efficiency of HTML crawling, we develop the *bounded HTML crawling* that holds some thresholds to limit search space. Again, this method bootstraps the seeding URLs for RDF crawling, and manual submission and inductive learner are involved in collecting the seeding URLs for HTML crawling.

- **RDF crawling.** *RDF crawler* enhances conventional HTML crawling by adding content validation and hyperlink extraction and selection components. It visits seeding URLs periodically to keep an up-to-date view of the Semantic Web. We employ several URL filtering heuristics to reduce the search space of RDF crawling. For example, the hyperlinks in RSS documents usually link to HTML documents, RDF crawler avoids such hyperlinks by skipping URLs that have filetype extension *html*. Its seeding URLs are collected from three sources: *meta crawling*, *bounded HTML crawling*, and itself.

IV.A.2 Sample Dataset

The harvest result of RDF crawling forms a global catalog of the Semantic Web. Moreover, we construct a sample dataset to evaluate the performance of these harvesting methods and to train the inductive learner.

A **sample dataset** consists of a set of observations. An **observation** consists of three parts: (i) a URL as the unique identifier; (ii) *labels* that shows the results of content validation; and (iii) *features* that shows the metadata of Semantic Web documents such as *ping state* (ping refers to connecting and downloading a Semantic Web document from the given URL).

[1] Semantic Web documents that populate classes in PML ontology.

The basic *label* of an observation can be one of the following:

- *SWD*, indicating a confirmed Semantic Web document. Moreover, we are interested in two non-exclusive sub-labels of SWD: (i)*SWO*, indicating Semantic Web Ontologies that contain definition of Semantic Web terms, (ii) *PSWD*, indicating pure SWDs that can be successfully parsed without warnings or errors.
- *NSWD*, (non Semantic Web document) indicating a confirmed conventional Web document;
- *UNKNOWN*, indicating download failure or content validation interruption.

The *features* of an observation are collected from the following sources:

- *text content*. For each keyword in the cached text content of the observation, we generate a feature to indicate its existence. Since SWDs are overwhelmed by conventional Web documents, we only cache the text content of all SWDs and some relevant NSWDs[2]; therefore, some observations labeled by NSWD may not have such keyword features extracted from text content. In order to leverage the feature space, we focus on numbers, letters and several punctuations such as ":", "-", "." and "_", and we treat the rest characters as white-space[3].
- *URL*. For each URL of an observation, we collect the following features like Google does: (i) filetype, i.e., the filetype extension; (ii) inurl, i.e., keyword extracted by decomposing the URL; (iii) site, i.e., domain name (including parent domain names) extracted from the host part of the URL. Moreover, we collect two extra features: *host*, i.e., the host part of the URL, and *base*, i.e., URLs being the parent paths of URL.
- *system metadata*. One candidate URL can be discovered by multiple methods; however, we only record the first pair of (method, seed) that discovers the URL. A **seed** could be a query string for Google crawling or a URL for HTML crawling or RDF crawling.

In practice, we prune highly repetitive observations (e.g., observations from `www.livejounal.com` and `onto.stanford.edu` where over 10,000 observations have been found having similar features) because they are usually automatically generated with predefined patterns and may overwhelm the *real signals*.

[2]we only cache the text content of some relevant NSWDs by heuristically checking the presence of RDF-indicators, e.g., "they are valid XML documents", "RDF namespace is a substring in their text", and "their URL contains substring *rdf*".

[3]This filter could be removed in the future once more computational resources are available.

IV.A.3 Evaluation Metrics and Inductive Learner

Based on the sample dataset, we quantify the performance of a pair of (method, seed) using three criteria:

- *true* criterion counts only the observations first discovered by the pair. A URL can be discovered by several pairs of (method, seed). Since these pairs are scheduled sequentially, the URL is *first discovered* by one pair and then *rediscovered* by other pairs. Only one pair of (method, seed) can first discover an observation. This criterion measures the actual performance of a pair.

- *would-be* criterion counts the observations that would have been discovered by the pair as long as they satisfy the constraints specified by the pair. This criterion provides an optimistic estimation on the performance of a pair.

- *google* criterion is an enhancement of *would-be* criterion. When employing Google-based meta-crawling, the number of observations that can be discovered by a pair is usually much larger than the estimation based on *would-be* criterion because our sample dataset skips indexing many relevant conventional Web documents (which should be labeled NSWD). Therefore, we need to adjust the estimation of the number of relevant observations of a pair by additionally considering Google's estimation to approximate the total number of relevant observations.

We use some metrics to measure the performance of a pair under the above three criteria. The general form of these metrics is

$$METRIC^{CRITERION}(l, m, s)$$

where $METRIC$ refers to the metric, $CRITERION$ can be any of the three criteria listed above, l refers to the label of an observation, m refers to a harvesting method, s refers to a seed of method m. Any of l, m, s can be wildcard "-", which refers to all possible cases. In this dissertation, we use the following metrics:

- *document frequency*. Since each observation can be mapped to a document, we refer document frequency $D^{CRITERION}(l, m, s)$ to the number of observations satisfying the given criterion and constraints. The following are example document frequencies based on different criteria:

  - $D^{true}(l, m, s)$ is the number of observations that have label l and are first discovered by the pair (m, s)
  - $D^{would}(l, m, s)$ is the number of observations that have label l and can be discovered by the pair (m, s)

  - $D^{true}(-, m, s)$ is the number of observations (regardless of the label) that are first discovered by the pair (m , s).
  - $D^{google}(-, G, s)$ is the adjusted[4] number of observations that can be discovered by querying Google with the seed s. In particular, G refers to *Google based meta-crawling*.

- *host count.* We use *host count* to study the discovery performance since SWDs from the same host are usually in similar patterns. Let $H^{true}(l, m, s)$ be the number of distinctive hosts of observations labeled by l and first discovered by the pair (m, s), and $H^{would}(l, m, s)$ be the *would-be* estimation.

- *Precision*. Equation IV.1 computes the *true* precision $Precision^{true}(l, m, s)$ as the percentage of observations labeled by l among all observations first discovered by (m,s).

$$Precision^{true}(l, m, s) = \frac{D^{true}(l, m, s)}{D^{true}(-, m, s)} \tag{IV.1}$$

  Similarly, we compute the precision for the "would-be" case using Equation IV.2, and the "google" case using Equation IV.3. We adjust the *would-be* estimation by adding Google's estimation: (i) the *would-be* estimation alone is not enough because out sample dataset has skipped many observations labeled NSWD, (ii) the Google's estimation alone is not enough because Google neither indexes the entire Web nor covers all URLs in our sample dataset.

$$Precision^{would}(l, m, s) = \frac{D^{would}(l, m, s)}{D^{would}(-, m, s)} \tag{IV.2}$$

$$Precision^{google}(l, G, s) = \frac{D^{would}(l, G, s)}{D^{would}(-, G, s) + D^{google}(-, G, s)} \tag{IV.3}$$

- *Recall*. Equation IV.4 computes the percentage of the observations labeled by l and first discovered by (m, s) among all observations labeled by l. Similarly, we compute the recall for the *would-be* case.

$$Recall^{true}(l, m, s) = \frac{D^{true}(l, m, s)}{D^{true}(l, -, -)} \tag{IV.4}$$

- F_1 *measure*. F_1 measure (see Equation IV.5) [137, 161] is a single value measure that reflects the tradeoff between precision and recall. We use it to compare the overall harvesting performance of a

[4]We adjust the *would-be* estimation by Google's estimation

pair. Similarly, we may compute the "would-be" and "google" F_1.

$$F_1^{true}(l, m, s) = \frac{2 \times Precision^{true}(l, m, s) \times Recall^{true}(l, m, s)}{Precision^{true}(l, m, s) + Recall^{true}(l, m, s)} \quad \text{(IV.5)}$$

The inductive learner uses the sample dataset to simulate *Google based meta-crawling* and *HTML crawling* and thus generate seeds for *Google base meta-crawler* and *bounded HTML crawler* based on the value of F_1 measure attached to those candidate seeds. The learner also uses the sample dataset to schedule *RDF crawling*. In our practice, the inductive learner is not a standalone program; instead, all the three automated crawlers have inductive learning components. We will discuss inductive learner in the rest of this chapter.

IV.A.4 Harvesting Strategies

In order to utilize the limit computation resources to harvest significant sample dataset of the Semantic Web, we adopt the following harvesting strategies to schedule crawlers' behaviors.

First, we prioritize harvesting URLs by their provenance:

- URLs from newly discovered websites should be visited in high priority to maximize the diversity of the harvesting results.
- URLs from giant websites in which more than 10,000 SWDs have been found should be visited or revisited in low priority because they seldom contribute new Semantic Web data usage patterns.

Second, we prioritize revisiting Semantic Web documents by their labels:

- *Semantic Web ontologies* are the most important because they are critical for users to encode and understand Semantic Web data.
- *pure Semantic Web documents* are the second important because they contain large amount of Semantic Web data and can be thought of as the building block of the Semantic Web.
- *embedded Semantic Web documents* are the least important because they usually contribute small amount of annotative Semantic Web data.

IV.B Google based Meta Crawling

We choose *Google based meta crawling* for several reasons: (i) it has indexed the largest number of Web documents among existing web search engines [67]; (ii) it does not filter Semantic Web documents out of search results; (iii) it provides a Web API which is friendly to meta-crawler; and most importantly (iv) it allows queries on both the text content and the URL of Web documents. In particular, we found the following primitive Google's constraints useful in harvesting candidate URLs of SWDs:

- **text pattern.** Text pattern search is the minimum requirement for conventional web search engines. Google search barely filters stop-words, for example, the query "a" returns 23,280,000,000 results. Therefore, query 1 in Table IV.1 can be used to estimate the upper bound of total number of online SWDs. Google search is case-insensitive, e.g., "rdf" and "RDF" return the same amount of result. It treats some punctuations (e.g., "-", ":") as hyphens of word-sequence. Therefore, query 2, 3 and 4 in Table IV.1 return similar numbers of results.

- ***filetype* constraint.** Google parses a URL to extract its filetype extension. It forbids the most generic filetype search, i.e., "filetype:rdf"; however, we may approximate this query by "filetype:rdf rdf". Moreover, we may combine this constraint with other constraints[5]. For example, query 5, 6 and 7 in Table IV.1 demonstrate the usage of *filetype* constraint.

- ***site* constraint.** Google can search URLs by the domain name of their host part: it can return URLs from the specified website. Although this constraint alone does not help users to find more Semantic Web documents, it has non-trivial contributions in practice. It is especially useful in the following use-cases: (i) since Google API only returns at most 1000 results for any query string submitted by the public users, site constraint helps users to refine the query into several sub-queries with disjointed results and thus to retrieve more results than Google's limit; (ii) only a few websites publish Semantic Web documents, site query can help us to focus on these sites. For example, query 8, 9 and 10 in Table IV.1 demonstrate the impacts of refining *site* constraint.

- ***inurl* constraint.** Google also indexes text patterns of URLs. This constraint is more generic than filetype and site query, and it is especially useful when a Semantic Web document is dynamically generated, e.g., by PHP or ASP. For example, query 11, 12 and 13 in Table IV.1 illustrate that the *inurl* constraint can help us find more URLs in addition to the results of *filetype* or *site* constraints.

[5]Obviously, no two filetype constraint should be connected by AND operator in one query string.

Table IV.1: Example Google queries and their estimated total results

id	query	# of results (as of March 18, 2006)
1	rdf	192,000,000
2	rdf-rdf	908,000
3	rdf:RDF	905,000
4	rdf.rdf	905,000
5	filetype:rdf	forbidden
6	rdf filetype:rdf	5,580,000
7	rdf filetype:owl	28,600
8	rdf site: edu	959,000
9	rdf site:umbc.edu	59,300
10	rdf site:cs.umbc.edu	317
11	inurl:rdf	26,400,000
12	inurl:index.rdf	3,850,000
13	inurl:22-rdf-syntax-ns	245

IV.B.1 Implementation

We have developed an efficient *GoogleCrawler* based on Google API with three desired features: (i) it usually generates efficient Google query strings, i.e., the percentage of SWDs in their results are significantly high; (ii) it schedules query operations efficiently by promoting good query string; and (iii) it is automated and requires minimum manual involvement.

As a regular client of Google API, we should efficiently use its limited query resources: (i) within 24 hours, a public user can submit up to 1,000 queries (each has two parts: the query string and the starting point of query result) and receive up to ten results for each query; therefore, at most 10,000 URLs can be returned a day. (ii) Google only returns the first 1,000 results for each query string when it has found more than 1,000 results. Even though insiders from Google could have avoided these limitations, the scheduling strategies described in this section is still useful in optimizing resource allocation.

The pseudo-code of *GoogleCrawler* is listed below. Two lists of Google queries are used: the primary Google query seeds G_{seed}, and the candidate Google query pool G_{pool}.

1. process all new queries in G_{seed}
2. revisit some best old queries in G_{seed}
3. update Google-estimated-total of some best queries in G_{pool}
4. generate new queries into G_{seed} from G_{pool}
5. GOTO step 1 until the daily limit of Google API has been reached
6. generate new queries into G_{pool}

In step 1, new queries in G_{seed} are always in the highest priority due to our preference in discovery. New queries are processed in first-come-first-serve manner based on the time when they have been added to the system. Note that once the daily limit of Google API has been reached at step 1 through step 5, we will jump directly to step 6.

In step 2, once all new queries have been visited, we may revisit some old queries in G_{seed}. Google dynamically updates PageRank of Web documents to reflect its observed changes of the Web. Hence, the first 1,000 results of a query may change over time, and we may revisit the same query to obtain new URLs. In order to estimate the performance of revisiting a Google seed, we use *discRatio* (see Equation IV.6 where $GoogleAPIcall(seed)$ is the number of Google API calls used on the seed) ,

$$\text{discRatio}(l, seed) = \frac{D^{true}(l, G, seed)}{GoogleAPIcall(seed)} \tag{IV.6}$$

This *discRatio* balances resource allocation and promotes discovery of new target URLs by allocating Google API calls to seeds that discover more new URLs using less API calls. In practice, we select a few seeds in G_{seed} for harvesting SWOs (see Table IV.2) and PSWDs (see Table IV.3) respectively. Moreover, an old seed is selected if it has the best *discRatio*, has first discovered at least one target SWD, and has not been visited for at least ten days. When no seeds meet the criteria, no seed will be selected.

Table IV.2: Top five Google seeds (out of 181) in harvesting SWOs

disc-ratio	seed	D^{true} (swo)	#call (google)	Pre^{true} swo
4.00	site:w3c.org filetype:owl	8	2	0.889
2.38	site:www.iis.uni-stuttgart.de filetype:rdf	429	180	0.625
1.74	site:svn.mindswap.org filetype:rdf	191	110	0.659
1.34	site:lojjic.net filetype:rdf	156	116	0.994
1.08	rdf site:countries.eea.eu.int	86	80	0.453

Table IV.3: Top five Google seeds (out of 521) for harvesting pure SWDs

disc-ratio	seed	D^{true} (pswd)	#call (google)	Pre^{true} pswd
4.00	site:w3c.org filetype:owl	8	2	0.889
4.00	rdf site:linuxintegrators.com	16	4	1.000
3.61	site:www.iis.uni-stuttgart.de filetype:rdf	649	180	0.946
3.39	rdf site:rdf.elsie.org.uk	78	23	0.975
3.27	rdf site:publish.pots.com.tw	144	44	0.713

In step 3, we spare some Google API calls to retrieve Google's estimation on the total number of Web documents matching a query. With this number, we can then compute "google" F_1 for candidate seeds in G_{pool}. Moreover, we can avoid emitting the candidate seeds to G_{seed} when Google API returns zero. In practice, we update ten best candidate seeds based on their "would-be" F_1 in finding SWOs and pure SWDs respectively. Note ten candidate seeds are chosen unless their Google's estimation has not been updated for at least 30 days. Table IV.4 lists ten best "would-be" candidates for harvesting SWOs in G_{pool} and their "google" F_1. By updating Google's estimation, some "would-be" queries are removed due to their low "google" F_1. For example, the query results for "xmlschema" are removed since most of them are XML documents but not SWDs. The last column "in-G_{seed}" indicates if the candidate seed from G_{pool} is already included in G_{seed}, and "1" means yes.

Table IV.4: Top ten candidate Google seeds for SWOs in G_{pool} ordered by "would-be" F_1

F_1^{would} swo	candidate seed	D^{would} swo	#total (pinged)	#total (google)	F_1^{google} swo	in G_{seed}
0.666	owl:class	9,835	10,804	106,000	0.145	1
0.575	subclassof	8,930	12,355	250,000	0.064	0
0.525	rdfs:comment	8,887	15,110	136,000	0.105	1
0.506	owl:ontology	7,833	12,261	118,000	0.105	1
0.494	rdfs:subclassof	7,001	9,637	143,000	0.082	0
0.479	xmlns:owl	11,646	29,877	37,400	0.271	1
0.471	xmlschema	9,634	22,162	13,600,000	0.001	0
0.451	objectproperty	5,687	6,514	167,000	0.059	0
0.445	rdfs:range	5,964	8,065	117,000	0.083	0
0.444	rdfs:domain	5,998	8,320	111,000	0.087	0

In step 4, we add some candidate seeds to G_{seed}. In practice, two best candidate seeds for finding SWOs are chosen if their "google" F_1 is greater than 0.01 and they are from $G_{pool} - G_{seed}$. Similarly, two best candidate seeds for finding PSWD are chosen. Table IV.4 shows the difference between the "would-be" F_1 and the "google" F_1, and many candidates with high "would-be" F_1 do not necessarily yield high "google" F_1, e.g., "xmlschema". Table IV.5 shows the ten best candidate seeds (for finding SWOs) that have been added to G_{seed} due to their high "google" F_1. This seed selection heuristics favors low "Google estimation", which partially implies that (i) SWDs are not overwhelmed by NSWDs in Google query results and (ii) Google has limitations in covering the entire Web.

In step 6, besides several initial seeds manually generated for bootstrapping, over 90% candidate seeds are automatically generated thereafter based on the growing sample dataset. Currently, two classes of query generation heuristics are used.

Table IV.5: Top ten candidate Google seeds for SWOs ordered by "google" F_1

F_1^{google} swo	candidate seed	D^{would} swo	#total (pinged)	#total (google)	F_1^{would} swo
0.398	owl:class filetype:owl	4,723	4,828	173	0.401
0.364	site:iw.stanford.edu filetype:owl	4,249	4,343	252	0.368
0.351	xmlns:owl filetype:owl	5,021	9,335	531	0.358
0.343	rdf:rdf filetype:owl	5,027	9,738	887	0.353
0.343	xmlns:rdf filetype:owl	5,027	9,741	888	0.353
0.333	rdf:resource filetype:owl	4,835	9,448	889	0.343
0.313	owl:ontology filetype:owl	3,842	5,497	325	0.317
0.278	rdfs:comment inurl:owl	3,477	5,469	857	0.287
0.276	xmlschema filetype:owl	3,715	7,722	491	0.281
0.271	xmlns:owl	11,646	29,877	37,400	0.479

- First, we may generate top-level candidate seeds, each of which has only one query pattern, using features in the sample dataset as long as they can be translated to Google query. In practice, we consider the following features: keyword of text content, filetype, site, and inurl. The features are further filtered by the following: their D^{would} should be greater than 100 for SWOs and 1000 for PSWDs.

- Second, we may further combine existing queries for achieving better performance as well as dealing with Google's limit, e.g., we may obtain more URLs than the first 1000 results if we refine a Google query by adding more constrains. This class of heuristics, in some sense, is essentially a refinement of the top-level queries, and is not necessary in the absence of Google's limit. To ensure their "would-be" performance and control the size of seed space, only a few best top-level candidates are chosen to participate combination. In practice, we have tried the following combination strategies to limit the search space: (i) keywords + filetype , (ii) keywords + site, (iii) keywords + inurl , (iv) keywords sequence, (v) inurl sequence, and (vi) site + filetype.

IV.B.2 Evaluation

By March 14, 2006, GoogleCrawler has committed 276,383 Google API calls and retrieved 2,403,792 URLs during the past 387 days. After filetype filtering, only 429,609 URLs from 26,431 distinctive websites have been saved. Among the 407,420 URLs being pinged, there are 5,879 confirmed SWOs, 121,431 confirmed PSWDs, and 193,799 confirmed SWDs.

Table IV.6 lists ten best seeds (out of 181) ordered by the number of SWOs found by them. The top three seeds were manually submitted one year ago, and they have accumulated SWOs using over 1,000 Google

calls. This observation also justifies the utility of *revisit* operations in meta-crawling. In fact, our seed generation mechanisms did re-discovered these queries using the sample data set. The rest six queries are automatically generated using heuristics that combines top-level Google queries.

Table IV.6: Top ten Google seeds contributing SWOs

D^{true} swo	seed	is auto	D^{would} swo	#total google	# API call google
1515	rdf filetype:owl	0	5,037	29,000	3,397
832	rdf filetype:daml	0	889	882	2,639
477	rdf filetype:n3	0	783	18,400	1,911
429	site:www.iis.uni-stuttgart.de filetype:rdf	1	456	809	180
312	rdf filetype:rdfs	0	540	295	508
191	site:svn.mindswap.org filetype:rdf	1	218	460	110
177	rdf site:www.schemaweb.info	1	179	919	607
156	site:lojjic.net filetype:rdf	1	374	616	116
120	rdf site:dublincore.org	1	274	18,800	901
117	rdf site:au.dublincore.org	1	120	628	320

Table IV.7 lists ten best seeds (out of 1234) ordered by the number of PSWDs contributed by them. The number of harvested PSWDs is usually lower than that estimated by Google because many URLs in Google search result have already been found by other harvesting methods and Google only returns the first 1,000 resulting URLs; however, seeds 1, 8, and 9 do collect more PSWDs than Google's estimation. We attribute the observation to two main resons: (i) revisiting a Google seed may yield new URLs as the result of Google Rank update; (ii) the estimated total returned by API is sometimes significantly lower than that returned by Google's web interface. Seeds 2, 3, and 7 discover more PSWDs using less API calls because they perform well in harvesting RSS documents.

Table IV.7: Top ten Google seeds contributing PSWDs

id	D^{true} pswd	seed	is auto	D^{would} pswd	#total google	#call google
1	5,322	rdf site:ws.audioscrobbler.com	1	5,639	883	3,544
2	4,478	rdf site:bulkfeeds.net	1	5,869	9,330	1,493
3	4,144	rdf site:yaplog.jp	1	9,241	2,560,000	2,669
4	3,499	rdf filetype:rdf xml -version jp	0	8	54,800	3,500
5	3,183	rdf site:blog.drecom.jp	1	6,649	106,000	3,316
6	2,654	rdf site:bitzi.com	1	2,713	17,800	2,807
7	2,612	rdf site:blog.goo.ne.jp	1	8,500	26,100	800
8	2,198	rdf filetype:rdf xml iso 8859	0	33,995	615	2,700
9	2155	rdf filetype:rdf xml -version -org edu	0	0	928	3,193
10	2,146	rdf site:blogs.dion.ne.jp	1	0	5,330	2,600

Table IV.8 and Table IV.9 list top five seeds generated by each type of heuristics for harvesting SWOs and PSWDs respectively. Two types of top-level candidate seeds have not yet been added to G_{seed} for different reasons: (i) candidate seeds generated by "site" heuristics have too low "google" F_1 (less than 0.1); (ii) candidate seeds generated by "filetype" heuristics has zero Google's estimated total because Google does not allow direct "filetype" query. We have the one interesting observation: *filetype* and *site* constraints are more effective than other constraints even though their *would-be* document frequencies are relatively low. This is because their *google* precisions are high.

Table IV.8: Best top-level Google seeds contributing SWOs

D^{true} swo	seed	is auto	D^{would} swo	#total google	#call google
heuristic: keyword					
26	xmlns:owl	1	11,646	37,400	200
8	rdf:property	1	4,537	108,000	100
6	inferenceweb	1	898	9,330	100
5	xmlns:cc	1	164	17,100	100
5	22-rdf-syntax-ns	1	18,596	1,040,000	100
heuristic: "inurl" constraint					
3	inurl:ontologies	1	2,034	8,310	209
heuristic: keyword + "filetype" constraint					
1515	rdf filetype:owl	0	5,037	29,000	3397
832	rdf filetype:daml	0	889	882	2,639
477	rdf filetype:n3	0	783	18,400	1,911
312	rdf filetype:rdfs	0	540	295	508
70	rdf filetype:rdf	0	4,128	5,510,000	2,701
heuristic: text pattern + "site" constraint					
177	rdf site:www.schemaweb.info	1	179	919	607
120	rdf site:dublincore.org	1	274	18,800	901
117	rdf site:au.dublincore.org	1	120	628	320
101	rdf site:knowledgeweb.semanticweb.org	1	205	168	94
98	rdf site:oaei.inrialpes.fr	1	124	189	164
heuristic: keyword + "inurl constraint					
19	owl:imports inurl:owl	1	2,611	568	115
15	xmlns:owl inurl:rdf	1	3,554	10,400	201
heuristic: "site" constraint + "filetype" constraint					
429	site:www.iis.uni-stuttgart.de filetype:rdf	1	456	809	180
191	site:svn.mindswap.org filetype:rdf	1	218	460	110
156	site:lojjic.net filetype:rdf	1	374	616	116
99	site:lists.w3.org filetype:rdf	1	123	418	596
30	site:w3.org filetype:n3	1	195	9,430	400
heuristic: multiple "inurl" constraints					
12	inurl:rdf inurl:ontologies	1	109	9,130	269
1	inurl:rdf inurl:owl	1	349	36,100	301
heuristic: keywords sequence					
3	xmlns rdfs	0	14,013	235,000	100
1	xmlns 1999 rdf ns	0	17,500	842,000	100

Table IV.9: Best top-level Google seeds contributing PSWDs

D^{true} swo	seed	is auto	D^{would} swo	#total google	#call google
heuristic: keyword					
99	xmlns:admin	1	9	28,500	100
54	xmlns:foaf	1	791	52,300	100
35	xmlns:owl	1	11,646	37,400	200
17	dc:language	1	832	788,000	100
12	xmlns:taxo	1	0	1,340	64
“inurl” constraint					
3	inurl:ontologies	1	1,974	8,310	209
2	inurl:rdf	1	236,344	28,900,000	204
heuristic: keyword + “filetype” constraint					
2121	rdf filetype:owl	0	8,723	29,000	3,397
1,000	rdf filetype:xml	0	4,305	168,000	2,727
965	rdf filetype:n3	0	1,833	18,400	1,911
868	rdf filetype:rdf	0	204,610	5,510,000	2,701
821	rdf filetype:nt	0	955	909	1,709
heuristic: keyword + “site constraint					
5,322	rdf site:ws.audioscrobbler.com	1	5,639	883	3,544
4,478	rdf site:bulkfeeds.net	1	5,869	9,330	1,493
4,144	rdf site:yaplog.jp	1	9,241	2,560,000	2,669
3,183	rdf site:blog.drecom.jp	1	6,649	106,000	3,316
2,654	rdf site:bitzi.com	1	2,713	17,800	2,807
heuristic: keyword + “inurl” constraint					
109	xmlns:owl inurl:rdf	1	14,687	10,400	201
15	owl:imports inurl:owl	1	3,428	568	115
heuristic: “site” constraint + “filetype” constraint					
1,490	site:blogs.dion.ne.jp filetype:rdf	1	10,049	6,280	1,400
1,458	site:yaplog.jp filetype:rdf	1	9,053	12,400	537
1,324	site:blog.livedoor.jp filetype:rdf	1	31,741	57,800	1,000
1068	site:blog.drecom.jp filetype:rdf	1	6,470	34,600	1,700
932	site:www.wasab.dk filetype:rdf	1	9,189	582	812
heuristic: multiple “inurl” constraints					
15	inurl:rdf inurl:ontologies	1	292	9,130	269
1	inurl:rdf inurl:owl	1	3,300	36,100	301
heuristic: keywords sequence					
152	xmlns 1999 rdf ns	0	337,597	842,000	100
5	xmlns rdfs	0	115,787	235,000	100

IV.C Bounded HTML Crawling

Based on the observed Power Law distribution of the number of Semantic Web documents per website collected from the sample dataset (see chapter V), we hypothesize Semantic Web documents tend to be clustered on some websites. Therefore, we may use HTML crawling to harvest the Semantic Web documents clustered on a few giant websites (hosting huge amount of SWDs). By choosing the appropriate starting points where SWDs are expected within several hops of crawling, the harvesting accuracy may be substantially above average. This method complements meta-crawling because it collects URLs by Web crawling.

The *bounded HTML crawling* is especially useful when the surfing paths between Semantic Web documents consist of several conventional Web documents. The *bounded* modifier bounds the computation resources by limiting the starting point and the search-space of HTML crawling to guarantee crawling performance. It employs several thresholds to guarantee the computation efficiency (i.e., high discovery rate and low computational cost) and the termination of crawling. The max crawling depth is five. During crawling, it tests the first 500 URLs, continues crawling unless the percentage of confirmed SWDs is higher than 5%, and stops crawling when it has exhausted all URLs or visited 10,000 URLs.

IV.C.1 Implementation

The operation logic of *bounded HTML crawling* has two aspects: (i) generating seeds and scheduling seed visit and (ii) running bounded HTML crawling from a selected seed.

The HTML crawler schedules seed visit by iteratively running the following steps:

1. *process all newly submitted seeds.* New seeds come from two sources: users' submission and automatically generated submission by the inductive learner.
2. *revisit old seeds.* Once all newly submitted seeds have been processed, K old seeds will be selected by their "true" precision. In practice, we set $K = 2$ and choose seeds that have the best "true" F_1 values and have not been crawled for at least 30 days.
3. *generate new seeds.* A constant number K of new seeds with the best "would-be" F_1 values according to our sample dataset will be added to the pool of crawling seeds.

Once a seed has been chosen, we conduct HTML crawling using bounded depth-first search strategy. The depth-first strategy is chosen because Semantic Web documents are not always directly interconnected or directly linked by an index page; instead, they are often linked by a root page through several hops of

hyperlinks and some nodes in the path are Web documents. For example, the *California Invasive Species Information Catalog*[6] has published many Semantic Web documents as alternative views to conventional HTML pages. Similar cases are websites hosting large number of generated SWDs such as Inference Web's proof directory[7], as well as "hub" Web pages that link to many SWDs such as DAML ontology library[8]. In these cases, the Depth-First-Search (DFS) maintains consistent precision value while the Breath-First-Search (BFS) may reach low precision at the beginning.

The search space is bounded by the following criteria: (i) *crawling depth threshold*, i.e., it crawls Web documents at most K (=5) hops from the seeding URL while surfing the hyperlinks; (ii) *upper limit of search space*, i.e., it crawls at most 10,000 Web documents; (iii) *precision threshold*, i.e., it dynamically updates the minimum threshold of precision so that higher precision is required when more Web documents have been visited; and (iv) *URL pattern constraints*, i.e., it only visits URLs matching the specified URL pattern constraints or at most two hops from one matched URL.

The last criterion is specially designed to discover SWDs outside the seed's website, and it is useful in dealing with "hubs" like DAML ontology library.

IV.C.2 Evaluation

Based on our current statistics, we have collected 796 HTML crawling seeds, including 742 manually submitted ones and 54 automatically generated ones. The automated part is recently added to HTML crawling and thus contributes only a few seeds. A total of 305,908 URLs have been harvested, including 232,576 SWDs, 54,372 SWOs and 84,027 PSWDs. The high percentage of SWD is ensured by running content validation before adding newly discovered URLs to the pool of RDF crawling seeds. Currently, the automatically generated seeds have only found 1 SWD and 13 PSWDs because they are revisiting the websites where many URLs have already been harvested.

Table IV.10 lists top ten seeds (out of a total of 62) that harvest the most SWOs, and Table IV.11 lists top ten seeds (out of a total of 106) that harvest the most PSWDs[9]. As long as HTML crawling continuously harvesting new SWOs and SWDs, its role as the complement of Google meta-crawling is justified.

[6] `http://cain.nbii.org/crisis/crisiscat/`
[7] `http://iw.standford.edu/proofs/`
[8] `http://ontologies.daml.org/`
[9] The statistics is generated from 92,594 URLs discovered by bounded HTML crawling, including 41,444 SWDs, 2,313 SWOs and 9,466 PSWDs

Table IV.10: Top ten HTML-crawling seeds contributing SWOs

D^{true} swo	seed
1,094	http://gollem.science.uva.nl/cgi-bin/pl-cvsweb/unstable/Ontologies/
326	http://www.mindswap.org/2003/
174	http://www.w3.org/2001/sw/BestPractices/WNET/wnNounsyn_v7.owl
144	http://www.aifb.uni-karlsruhe.de/viewAIFB_OWL.owl
61	http://www.schemaweb.info/
48	http://orlando.drc.com/semanticweb/
47	http://dmag.upf.es/ontologies/
47	http://www.fruitfly.org/~cjm/obo-download/
46	http://62.149.228.132:8080/platypus/introspector/
39	http://www.mindswap.org/2004/

Table IV.11: Top ten HTML-crawling seeds contributing PSWDs

D^{true} pswd	seed
1,964	http://semanticweb.org/
1,851	http://www.mindswap.org/2003/
1,075	http://semspace.mindswap.org/2004/
996	http://gollem.science.uva.nl/cgi-bin/pl-cvsweb/unstable/Ontologies/
487	http://projects.semwebcentral.org/owl-gforge/
383	http://semwebcentral.org/
338	http://www.rossettiarchive.org
254	http://teleformacion.ifes.es/
237	http://www.w3.org/2001/sw/BestPractices/WNET/wnNounsyn_v7.owl
193	http://journal.dajobe.org/journal/2003/07/semblogs/

IV.D RDF Crawling

Swooglebot is one of many Semantic Web crawlers [34, 5, 18]. Unlike conventional HTML crawlers, it does content valuation and selectively follows some links extracted through HTML parsing and RDF parsing. It distinguishes itself from other Semantic Web crawlers in that it adopts a breath-first-search harvesting strategy and its seeding URLs have been significantly enriched by the URLs automatically collected by Google based meta-crawling and bounded HTML crawling.

IV.D.1 Implementation

The operation logic of *RDF crawling* has two aspects: (i) scheduling seed visit and (ii) processing a Web document using HTML and RDF parsing.

Seeds are visited primarily by their priority. A seeding URL should be revisited to check its changes. When two seeds have the same priority, they are prioritized by their last visited time in first-come-first-serve

manner. Swooglebot schedules visiting different groups of seeds in decreasing importance:

1. confirmed SWOs
2. new seeds
3. confirmed SWDs from websites contributed less than 10,000 seeds
4. old seeds (e.g. NSWDs) from websites contributed less than 10,000 seeds
5. the rest seeds

Once a seeding URL has been chosen, the RDF crawler visit the URL and conduct the following operations:

1. *download content*. Two issues are notable: (i) we need to specify preferred content type in RDF request in the order of : *application/rdf+xml*, *text/xml* , and then */*; (ii) the URL of a namespace can be redirected, e.g., the namespace URL of the Dublin Core Element ontology, `http://purl.org/dc/elements/1.1/`, redirects to its real physical URL, `http://dublincore.org/2003/03/24/dces#`. A Semantic Web crawler must capture redirection and use it as an additional heuristic to discover URLs of SWDs.

2. *validate content.* An important issue during content validation is to determine the content encoding because wrong content encoding may result in unrecognizable characters. We determine the final charset (character set) used by the document from three sources: content-type field in http response, the first several bytes of the downloaded document, and encoding declaration specified in XML prolog. Another issue is to handle embedded SWDs, each of which embeds RDF graph in RDF/XML.

3. *extract seeds from content parsing.* We only extract HTTP protocol based URLs using the following heuristics: (i) an absolute URL, (ii) URLs indicated by hyperlink “href”, except automatic links generated by directory listing; (iii) redirection URLs in embedded *Javascript* scripts, and (iv) URLs extracted from “link” and “frame” markups in which “src” is used like “href”. Once all URLs have been extracted, a filetype filter will remove URLs which can be directly classified as NSWDs without content validation. This judgment is made by matching a list of stop-extensions. A *stop-extension* can be predefined, e.g., *gif* and *doc*, or learned from sample dataset as long as it is used by no more than one SWD but more than 100 NSWDs, e.g., “java” and “xpi”. Table IV.12 and Table IV.13 list top ten seeding URLs contributing the largest amount of SWOs and PSWDs respectively.

Table IV.12: Top ten RDF-crawling seeds contributing SWOs using content parsing

D^{true} swo	seed
51	http://owl.mindswap.org/2003/meta/meta.rdf
36	http://iw.stanford.edu/proofs/RamaziOwnsGourmetFoods/id_85.owl
16	http://www.atl.lmco.com/projects/ontology/ontologies/AlignmentTasks.n3
13	http://www.daml.org/ontologies/ontologies.daml
12	http://ontoware.org/cmi/portal.owl
10	http://www.ksl.stanford.edu/software/IW/ex/jtp/aquaint/meetwithns2_1.daml
10	http://www.ksl.stanford.edu/software/IW/ex/jtp/aquaint/meetwithns1_1.daml
10	http://gollem.science.uva.nl/cgi-bin/pl-cvsweb/~checkout~/unstable/Ontologies/Base/
9	http://200.217.149.102:8080/proofs/ProposeAndRevise_1_Verify_10ns1_34.owl
8	http://www.ksl.stanford.edu/software/iw/ex/jtp/laptop/ex9ns1_1.daml

Table IV.13: Top ten RDF-crawling seeds contributing PSWDs using content parsing

D^{true} pswd	seed
381	http://xml.mfd-consult.dk/foaf/explorer/?foaf=http://www.amk.ca/index.rdf
358	http://owl.mindswap.org/2003/meta/meta.rdf
219	http://ajft.org/rdf/ajft.rdf
194	http://www.wasab.dk/morten/2005/04/photos/fanoe/2/index.rdf
135	http://soft-pc3.zib.de/MathInd/fkz03HOM3A1/publicationsProb/overview.shtml.rdf
133	http://sw.deri.org/2005/01/pyscutter/blogs.rdf
126	http://msdneventsbloggers.net/Bloggers.foaf
114	http://SemSpace.mindswap.org/2004/meta/meta.rdf
87	http://www.w3.org/2001/sw/DataAccess/tests/
75	http://dannyayers.com/2004/11/roll.rdf

4. *extract seeds from RDF parsing.* We selectively follow links using some popular heuristics: (i) namespace of a SWT that links to the definitions of SWO, (ii) URIrefs of the instances of certain classes, e.g., *owl:Ontology*, and (iii) URIrefs extracted from triples using link-indicators, e.g., *rdfs:seeAlso* is a good link-indicator for surfing FOAF personal profiles. Table IV.14 lists the most frequently used link-indicators and their harvesting results. Currently, the indicators are only selected from RDFS, OWL and DAML ontologies.

Table IV.14: The most frequently used link-indicators

indicator	#swo	#pswd	#swd	#url
rdfs:seeAlso	64	323,966	362,574	1,336,031
owl:imports	141	159	173	272
rdfs:isDefinedBy	26	28	42	138
daml:imports	18	17	24	48
owl:priorVersion	5	5	5	14

IV.D.2 Evaluation

Based on our current statistics, the RDF crawler has found 1,614,249 URLs using three heuristics, namely, “u” for URL redirection, “e” for selectively following links extracted from text content, and “s” for selectively

following links extracted from parsed RDF graph. Table IV.15 compares the harvesting performance of the three heuristics. According to Table IV.14 the contribution made by the last heuristic is mainly the result of *rdfs:seeAlso*'s hyperlink usage in FOAF space. The observation that the second heuristic found more SWOs than the last one is mainly due to the usage of Inference Web ontology, where all PML documents contain a mix of SWT definitions and class-instances.

Table IV.15: Performance of three harvesting heuristics of RDF-crawling

heuristics	#swo	#pswd	#swd	#url
url redirection	198	9,739	13,543	37,315
content parsing	2,941	22,327	33,971	78,121
RDF parsing	1,176	325,843	387,086	1,498,813
-	4,315	357,909	434,600	1,614,249

IV.E Overall Evaluation

The current sample dataset *SW06MAR* is collected based on one year observation between Jan 2005 and Mar 2006. The sample dataset consists of 2,769,037 URLs (including 1,179,335 SWDs, 848,254 PSWDs, and 73657 SWOs). The composition of *SW06MAR* is shown in Figure IV.2.

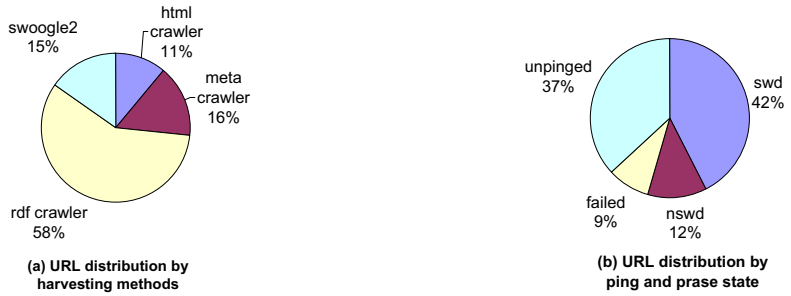


Figure IV.2: The composition of URLs in *SW06MAR* by harvesting methods and by ping state

- In Figure IV.2(a), we have manually added 419,271 previously collected URLs that are labeled by *Swoogle2* besides the three automated harvesting methods. *RDF crawler* has harvested huge amount of URLs mainly because of the popular usage *rdfs:seeAlso* in FOAF documents.
- In Figure IV.2(b), the hybrid framework exhibits high discovery accuracy 42% observations are labeled by SWD. Swoogle also skips harvesting the 37% candidate URLs (labeled by "unpinged") because of the balanced harvesting strategy.

Figure IV.3 provides a detailed analysis on the contribution of each harvesting method. The bars illustrate the harvesting methods' contributions to URLs in different ping states. *RDF crawler* have found the largest amount of URLs and confirmed SWDs mainly because of the popularity of FOAF applications. Meta crawler has the lowest harvest precision due to its discovery strategy. HTML crawler introduces minimum "failed" URLs because it has filtered such URLs before adding new URLs to the dataset.

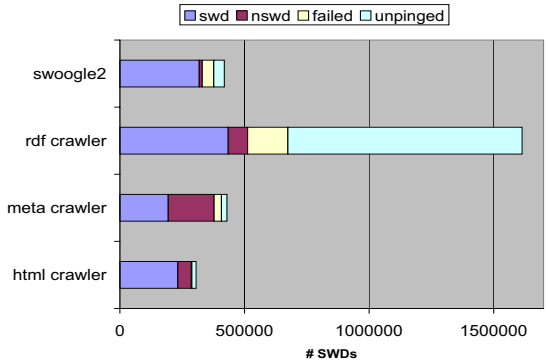


Figure IV.3: The detailed composition of URLs in *SW06MAR*.

Figure IV.4 illustrates the daily statistics of our crawling results. In general, the numbers of harvested SWDs, SWOs and PSWDs keep on growing. An interesting pattern in these curves is called *milestone*, where the "ping" curve touches the "url" curve. After this point, the number of URLs increases significantly. This observation is mainly caused by the strategy that delays harvesting URLs from websites hosting more than $10,000$ URLs until all other URLs have been visited. The figure also depicts the increasing gap between SWDs and the pinged URLs even though the absolute number of SWDs is still increasing. This is mainly because more heuristics with lower precision have been tried in discovering new URLs.

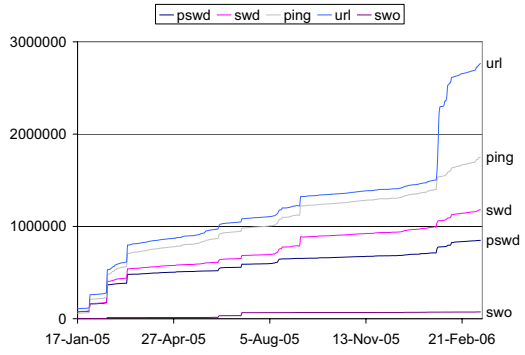


Figure IV.4: The daily harvest statistics between Jan 17, 2005 and March 12,2006

IV.F Summary

In this chapter, we elaborate a hybrid framework that integrates three automated methods in harvesting Semantic Web documents on the Web, namely Google based meta-crawling, bounded HTML crawling, and RDF crawling. The design and implementation of these methods are not completely new; instead, they are mainly common sense among Semantic Web researchers. The uniqueness of our harvesting methods is in fact the hybrid framework: (i) manual submission feeds seeding URLs to the *Google based meta-crawler* and the *bounded HTML crawler*; (ii) the meta-crawler and the HTML crawler automatically find and then feed seeding URLs to the *RDF crawler*, and (iii) the harvesting results of the RDF crawler can be used to inductively derive new seeding URLs for meta-crawler and HTML crawler.

These harvesting methods together build a sample dataset covering over one million SWDs. This number reasonably justifies the significance of the sample dataset even though it is far less than 11.5 billion of Web pages [67]. Moreover, the diversity requirement is enforced by the harvesting strategies: (i) although exhausting all SWDs is the ultimate goal, we argue that SWOs and pure SWDs should be harvested first. (ii) Newly discovered website should be visited in high priority while websites contributed over 10,000 URLs should be visited in lowest priority.

Chapter V

MEASURING THE SEMANTIC WEB

Two issues arise once we have harvested non-trivial number of Semantic Web documents: (i) the significance of our sample dataset, i.e. how well it samples the Semantic Web on the Web and approximates the corresponding global catalog, and (ii) the deployment status of the Semantic Web, i.e. the distribution of Semantic Web data calculated using our sample dataset.

V.A Significance of SW06MAR

Our sample dataset *SW06MAR* has 2,679,070 URLs, which include 1,179,335 SWDs from 157,003 distinct hosts. Among these SWDs, 73,657 can be classified as SWOs and 848,254 can be classified as PSWDs. The SWDs by themselves form a significant dataset [68]. Although the dataset is much smaller than the Web that has over 11.5 billion documents [67], it is significantly larger than datasets collected by related works.

- (2002) Eberhard [48] reported 1,479 SWDs with about 255,000 triples out of nearly 3,000,000 Web documents.
- (2003) OntoKhoj [127] reported 418 ontologies out of 2,018,412 Web documents after 48 hours crawling.
- (2004) DAML Crawler [34] reported 21,021 DAML files out of 743,017 Web documents as of April, 2004.
- (2004) Swoogle version 1 had collected 16,000 SWDs including 4,880 SWOs during May 2004.
- (2005) Swoogle version 2 had incrementally collected 328,987 SWDs including 11,278 SWOs out of 1,194,992 URLs between July 2004 and Feb 2005.

We additionally justify the significance of *SW06MAR* by (i) estimating the "real" size of the Semantic Web and (ii) estimating the coverage of SW06MAR.

V.A.1 Estimating the Size of the Semantic Web

The number of SWDs on the Web can measure the size of the Semantic Web. However, this number is hard to obtain because (i) Semantic Web documents are sparsely distributed on the Web and (ii) validating whether a Web document is a Semantic Web document requires non-trivial computation.

Drawbacks of Brute-force Approaches

Brute-force sampling methods widely used in measuring the size of the Web (e.g., testing 80 port for a huge list of IP addresses) [100] are not suitable due to their unacceptable low efficiency. In order to verify if a website contains SWDs, exhaustive crawling is needed to visit all web pages from the website until an SWD is confirmed. In addition, false results could be reached because a crawler cannot guarantee exhausting all Web documents on a website.

Drawbacks of Overlap Analysis on Meta-search

Researchers also estimate the size of the Web by measuring the overlap between the search results of conventional search engines [15, 67]. Unfortunately, conventional search engines are not suitable for indexing Semantic Web documents for three main reasons. (i) They seldom index markups that are heavily used by SWDs in encoding semantics. (ii) They rank SWDs lower because SWDs are not well linked by other Web pages and SWDs are less desired by human users. (iii) They intentionally avoid indexing SWDs that are dynamically generated by Web services. For example, even both support filetype search, only Google search but not MSN search supports searching filetype “rdf” and “owl”.

In order to measure how conventional search engines have indexed SWDs, we compare their search results using several queries. According to *SW06MAR*, 99% of SWDs have declared *RDF namespace*, whose URL is `http://www.w3.org/1999/02/22-rdf-syntax-ns#`, as non-markup which should be indexed by conventional search engines. Therefore, we use this URL to compose sevearl Google queries. Table V.1 lists six Google queries, each of which is associated with the estimated total number of search results reported by four popular web search engines, namely Google, Yahoo, MSN, and Ask.com.

- *query 1* is the entire URL.
- *query 2* is the longest word sequence when “/” is used in tokenization.
- *query 4* removes all numbers, stop-words and word sequence constraints.

- *query 5* is the most representative word in the URL.
- *query 3 and query 6* intend to filter HTML documents from the search results of *query 2* and *query 5* respectively.

Table V.1: The estimated number of results from four popular search engines (March 14,2006)

query	**Google**	**Yahoo**	**MSN**	**Ask.com**
1 (RDF namespace URL)	1	59,800	n/a	25,100
2 (22-rdf-syntax-ns)	1,010,000	499,000	78,847	28,900
3 (22-rdf-syntax-ns -inurl:html)	796,000	470,000 *	42,069	n/a
4 (rdf syntax ns)	1,400,000	492,000	60,868	31,600
5 (rdf)	220,000,000	25,300,000	1,967,786	3,392,000
6 (rdf -inurl:html)	108,000,000	25,700,000*	1,962,108	n/a

* Yahoo's inurl filter does not work properly in removing URLs containing "html".

We have some interesting observations: first, the estimated total number of estimated results returned by MSN and Ask.com in Table V.1 are usually 10 to 100 times less than Google. Our further analysis shows that they return fewer SWDs in the first 100 results. Second, the returned SWDs are mainly (i) N-Triple or N3 documents due to the non-trivial term frequency of RDF namespace, and (ii) RSS documents and Dublin Core Element ontologies due to their non-trivial in-links. Third, some query results may be dropped by "restrictive" queries, for example, *query 5* but not *query 3* can find the document `http://www.mozilla.org/news.rdf` using Google Search.

Therefore, only Google and Yahoo have indexed significant amount of SWDs, and removing Yahoo will not affect the *order of magnitude* (i.e., the smallest power of ten needed to represent a quantity [156]) of the estimated amount of indexable SWDs.

Bounding the Size of the Semantic Web using Google Query

We quantify the size of the Semantic Web by bounding two values: the number of indexable SWDs and corresponding *order of magnitude*.

We specially choose only *indexable SWDs* because no search engine can exhaust all documents on the Web, index all words and word sequences in a document, or return all relevant documents to a query due to resource and performance limitations. By cross-validating Google's results with *SW06MAR* ,which has already confirmed over 1,000,000 SWDs, we have observed that Google intentionally avoids indexing all Web documents (especially SWDs). For example, Google has indexed 30,200,000 Web documents from

livejournal.com according to the query "site:livejournal.com"; however, only 342 results can be retrieved by the query "site:livejournal.com inurl:foaf" while *SW06MAR* has indexed 100,517 SWDs from this website having URLs like `http://www.livejournal.com/users/USERNAME/data/foaf`.

We also classify SWDs into static SWDs and dynamically generated SWDs. Unlike the static SWDs, it is hard to measure the amount of SWDs dynamically generated due to the huge database behind the generator and additional combinatorial complexity. In fact, we have observed huge amount of SWDs dynamically generated by several websites[1]. Moreover, it is easy to write a dynamic Web page to generate unlimited number of SWDs each of which populates a unique instance of the class *Integer*. Therefore, when estimating the size of the Semantic Web we should be careful in counting dynamic SWDs and protecting the diversity of the Semantic Web. In our study, we consider bounding indexable SWDs, i.e., all static SWDs accessible on the Web and some dynamic SWDs in controllable amount.

To estimate the upper bound, we need a general Google query with high recall (retrieving most SWDs) and good enough precision (limiting the portion of NSWDs). Theoretically, *query 6* in Table V.1 should retrieve all SWDs since "rdf" is a common keyword in namespace declaration part. This query results in an estimation of 108,000,000 SWDs. However, our manual analysis shows that *query 6* does miss many RSS and FOAF documents, for example, "inurl:rss -rdf -filetype:html" returns additional RSS files. Therefore, we expand the query to a more complex one

rdf OR purl.org.dc OR inurl:foaf OR inurl:rss -filetype:html

to revise the estimated upper bound of the size of the Semantic Web. The query returns a higher estimation of 211,000,000 SWDs. Note both estimations indicate the upper bound of SWDs being in the order of 10^9.

To estimate the lower bound, we need a specific Google query with high precision (most retrieved URLs are SWDs) and good enough recall (covering non-trivial amount of SWDs). We compute the lower bound by summing up static SWDs having popular Semantic Web filetype extensions. Since we are computing lower bound, skipping files without filetype extensions only decreases the precision of but not nullifies the result. Additionally, most SWDs without filetype extensions are dynamical ones and should not affect the lower bound; and those well-known SWOs without filetype extensions are in trivial amount. Table V.2 lists popular filetype extensions inductively learned from *SW06MAR* with their document frequency and precision. Since Google forbids direct queries like "filetype:rdf", we collect estimated total by adding the extension as text keyword since it always appears in the URL, e.g., we use "owl filetype:owl" to find the total number of

[1]For example, liverjournal.com has generated a FOAF document for each of its 10 million users `http://www.livejournal.com/stats.bml`, Mar 15 2006).

documents having “owl” as file extension (the “+self” column in Table V.2 is based on this heuristic). In most cases, the file extension alone may produce biased results, so “rdf” is used to protect precision. For example, we have found many RSS files (e.g., version 0.91) which do not produce any triple, hence “rdf filetype:rss” is chosen instead of “rss filetype:rss” because of its high precision (see the “+rdf” column in Table V.2). Therefore, we have reached an estimation of at least 5.4 million SWDs and get the lower bound of SWD in the order of 10^7.

Table V.2: Popular SWD extensions in *SW06MAR* and their estimated total by Google

	SW06MAR		Google	
	total	precision	+ self	+ RDF
filetype:rdf	267,292	0.80	5,170,000	5,170,000
filetype:owl	61,739	0.73	55,000	28,900
filetype:rss	34,016	0.86	5,690,000	46,600
filetype:xml	4,329	0.09	71,300,000	170,000
filetype:n3	2,585	0.41	34,100	18,100
filetype:foaf	1,478	0.95	10,100	62
filetype:daml	1,376	0.32	17,900	851
filetype:nt	1,268	0.55	26,700	910
filetype:xrdf	573	0.94	275	195
filetype:bak	564	0.98	322,000	218
filetype:rdfs	526	0.73	515	309
total	375,746	n/a	82,626,590	5,436,145

V.A.2 Evaluating Coverage of Sample Dataset

We evaluate the coverage of our sample dataset by comparing SWDs per website with Google’s estimation. Figure V.1 depicts the number of SWDs for 1,239 websites each of which hosts no less than 10 confirmed PSWDs. For each website, the number of SWDs in *SW06MAR* from that website is plotted as a dot in “sw06mar” curve, and Google’s estimated total number of SWDs from that website[2] is plotted as a dot labeled by “google estimate”. For example, we use Google query “rdf OR purl.org.dc OR inurl:foaf OR inurl:rss -filetype:html site:www.cs.umbc.edu” to get Google’s estimated total SWDs from the website “www.cs.umbc.edu”. Each unit on x-axis in Figure V.1 corresponds to a website, and the 1,239 websites are sorted by *SW06MAR*’s estimation in descending order.

It is easy to see that the distribution of the corresponding Google’s estimation, even with high variance, does exhibit similar trend to the distribution of *SW06MAR*’s estimation. We attribute the variance to three

[2] the number is obtained from a Google query that concatenates the query string for deriving upper bound and the “site” constraint on the website

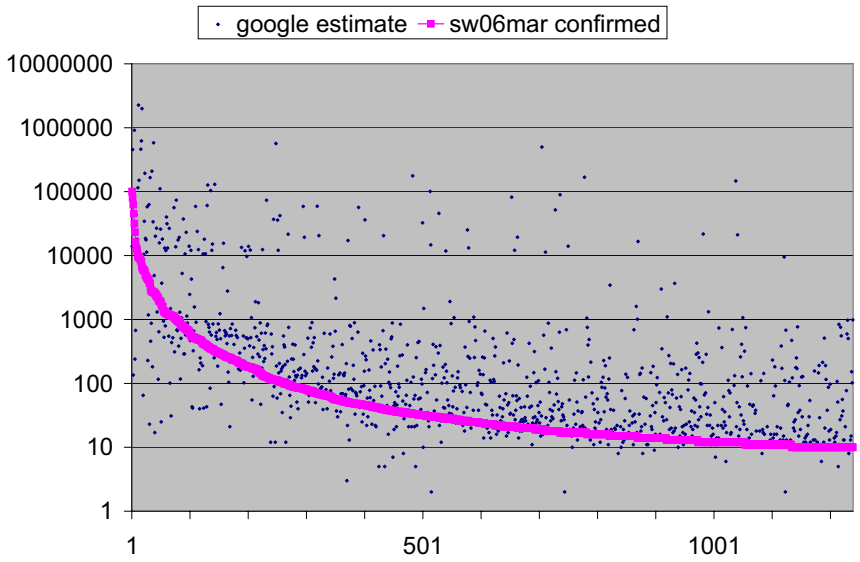


Figure V.1: The number of SWDs per website estimated by *SW06MAR* and Google.

main sources. (i) Google's estimation is too optimistic due to its heavy usage of "OR" operation; therefore, a Web document having "rss" in URL is not necessarily an SWD. (ii) Google site query searches all sub-domains of the site name (e.g., site:w3.org also returns results from www4.w3.org); on the contrary, *SW06MAR*'s results are exclusively partitioned. (iii) *SW06MAR* is collected using efficient harvesting methods so that it may index less SWDs (see dots above the curve) because our crawlers use far less harvesting seeds than Google and kept a long to-crawl list, and it may index more SWDs (see dots below the curve) because it complements Google's crawling limitation.

By summing up the number of SWDs of the 1,239 websites in Figure V.1, "google estiamte" results in 12,647,884 and "swd06mar confirmed" results in 697,333. Although the latter is only about 5.5% of the former, the latter has more URLs than Google API could return[3] and the former may have many duplicate URLs. The 848,254 confirmed PSWDs in *SW06MAR* cover at least 0.08% of the Semantic Web given the upper bound being 10^9, and at best 8.48% of the Semantic Web given the lower bound of the Semantic Web being 10^7.

[3]We summed up all possible results could be returned by Google API based on its 1000-URL limitation and resulted in 285,914 URLs in total for the 1,239 websites.

V.B Measuring Semantic Web Documents

SWDs are the atomic containers for transferring Semantic Web data and the interfaces between the Web and the RDF graph world. Therefore, we use the statistics obtained from *SW06MAR* to measure the interesting properties of SWDs. In particular, we are more interested in pure SWDs and SWOs.

V.B.1 Source Websites of SWD

In order to measure how Semantic Web data is distributed on the Web, we group SWDs by their source websites. Figure V.2 depicts the distribution of the number of websites[4] that host more than m pure SWDs collected by August 2005 (Figure V.2-b) and by Mar 2006 (Figure V.2-a) respectively.

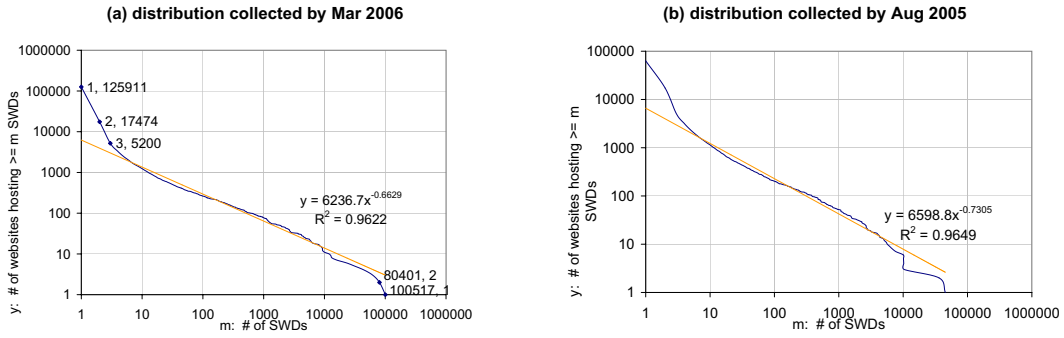


Figure V.2: The cumulative distribution of the number of SWDs per website

Both distributions can be perfectly regressed to Power distributions with similar parameters as shown by the close to one R^2 (R-square) value. Hence, we can hypothesize invariant cumulative Power distribution of the number of SWDs per website in the real Semantic Web.

Both distributions are skewed by two types of websites. First, the tail of the curve has a sharp drop when approaching 100,000 in x-axis. This shape is caused by our balanced harvesting strategy that delays harvesting SWDs from giant websites that have already contributed 10,000 URLs. Second, the head of the curve also has a sharp drop due to virtual hosting technology: some content publishing websites automatically offer each of its users a unique virtual host name under its domain. The two types of websites have contributed huge amount of dynamically generated SWDs that have similar content patterns; hence, they could overwhelm other interesting content patterns in the Semantic Web.

Table V.3 shows ten source websites hosting the largest number of pure SWDs. The “content” column

[4] A website is uniquely identified by its URL (host name) but not its IP.

shows the content topic of the SWDs from each website, such as personal profiles (i.e., FOAF documents), personal blog, RSS feed documents, portable proofs (PML documents) and publication information. The "un-pinded" column shows that we have skipped crawling those websites to preserve the diversity of *SW06MAR*.

Table V.3: Ten largest source websites of PSWDs

website	#PSWDs	unpinged	content
www.livejournal.com	100,518	79,331	foaf
www.tribe.net	80,402	25,151	foaf
www.greatestjournal.com	62,453	835	foaf
onto.stanford.edu	45,278	206	pml
blog.livedoor.jp	31,741	6,733	foaf
www.ecademy.com	23,242	3,281	foaf
www.hackcraft.net	16,238	0	dc, book
www.uklug.co.uk	13,263	2	rss
users.livejournal.com	12,783	40,211	foaf
ch.kitaguni.tv	11,931	3,010	rss

V.B.2 Size of SWD

The size of a SWD indicates the volume of Semantic Web data in the SWD, which is usually measured by the number of triples in the RDF graph parsed from the SWD. Figure V.3 depicts the distribution of SWDs' size: Figure V.3-a shows the distribution of the number of SWDs hosting exactly m triples; and V.3-b shows the distribution (Power distribution again) of the number of SWDs hosting more than m triples.

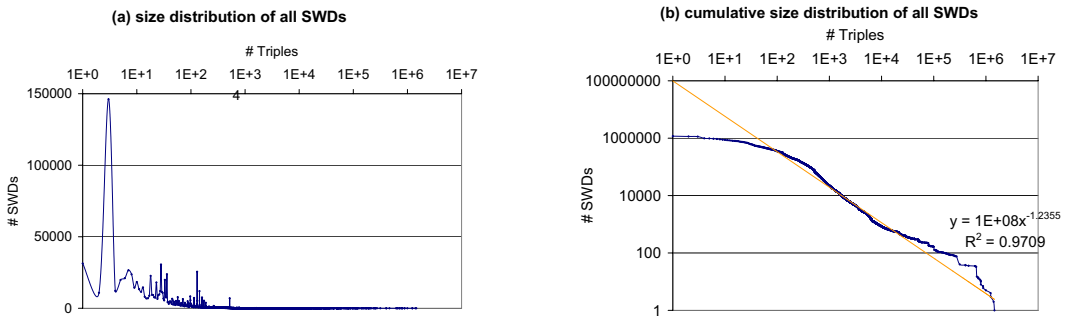


Figure V.3: The distributions of the number of triples per SWD

We further investigate the sub-classes of SWD, namely, embedded SWD, pure SWD and SWO to find interesting patterns, and derive Figure V.4.

- In Figure V.4-a, embedded SWDs are usually in small size: 69% have exactly 3 triples and 96% have

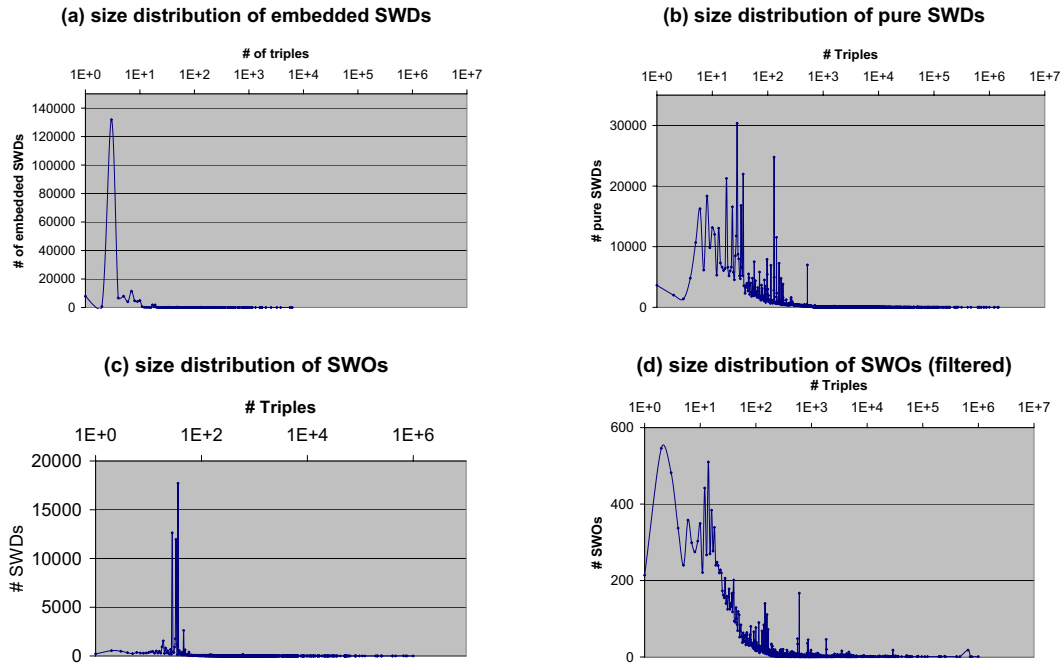


Figure V.4: The distribution of the number of triples per ESWD (or PSWD, or SWO)

no more than ten triples.

- In Figure V.4-b, the size of PSWDs varies: 60% have 5 to 1000 triples. The unusual peaks in the curve are caused by several special types of automatically generated SWDs which publish Semantic Web data in fixed patterns. For example, many PML documents have exactly 28 or 36 triples, and many RSS documents have exactly 130 triples[5].

- In Figure V.3-d, the distribution of the sizes of SWO is based on the removal of tens of thousands of PML documents (the original distribution is shown in V.3-c). It is interesting to find that over 1,000 SWOs have no more than three triples, and our further investigation shows that they are mainly testing data from e.g., RDF test and OWL test. Another interesting observation is that the size of SWO could be as large as 1 million triples[6].

[5] A RSS file usually publishes 1 instance of *rss:channel* with 8 description triples, 15 latest news items each of which has 7 description triples, and 1 rdf:seq instance contributing 17 triples which connect the instance of *rss:channel* to the instances of news items. Many such RSS files are automatically generated by a popular Blog publishing software "movabletype".

[6] `http://www.fruitfly.org/~cjm/obo-download/obo-all/ncbi_taxonomy/ncbi_taxonomy.owl` is the largest ontology we have found so far. Its file size is 140,410,529 bytes and it has 996,222 triples defining 291,797 classes and properties.

V.B.3 Top-level Domains of SWD

We also study the sources of the Semantic Web data using top-level domain extracted from the URLs of SWDs. Figure V.5 shows nine most used top-level domains ordered by the number of websites (hosts) and the "other" entry for the rest domains. Pure SWDs dominate the data space of the Semantic Web. "com" has contributed the largest portion of hosts (72%) and pure SWDs (43%) because of industry adoption of RSS and FOAF. The largest portion of SWOs is contributed by "edu" domain (38%)[7] immediately followed by "org" (21%) because of strong interest in developing ontologies in academia.

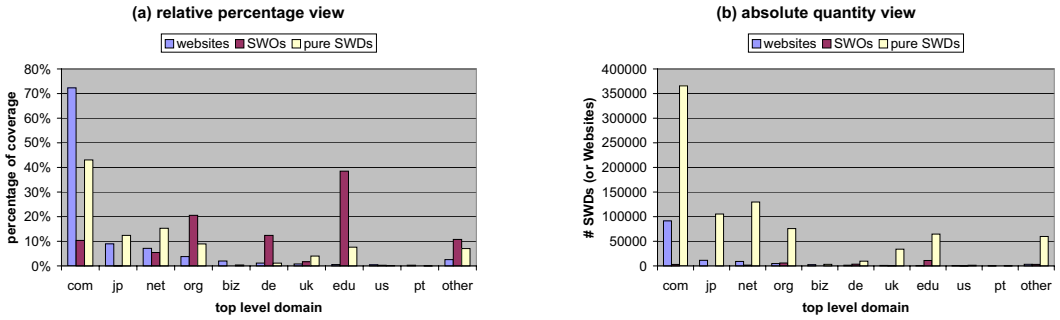


Figure V.5: Top-level domains used by SWDs

V.B.4 Age of SWD

SWDs could be uploaded, modified and removed on the Web. We measure the age of an SWD by the last-modified time (attached in the header of HTTP response) of its latest version. Figure V.6 plots the number of PSWDs and SWOs having been last-modified before the year-month on X-axis. The plot excludes PSWDs that do not have last-modified time specified in HTTP header.

The "pswd" curve exhibits exponential growth; intuitively, the growth implies that either many new PSWDs have been added to the Semantic Web or many old PSWDs have been updated recently. This statistics supports the hypothesis that the Semantic Web is growing rapidly.

The "swo" curve is plotted after filtering PML documents. The growth of SWOs has an increasing trend but the growth rate is slowing down. Intuitively, we may guess the trend in the emergence of the Semantic Web: active ontology development in the early time and transition to reusing ontologies in recent years.

We also notice the death of an SWD's version. Updating an SWD leads to the death of the old version and

[7] The result excludes the 45,278 PML documents from *onto.stanford.edu:8080* that are not intended to be ontologies.

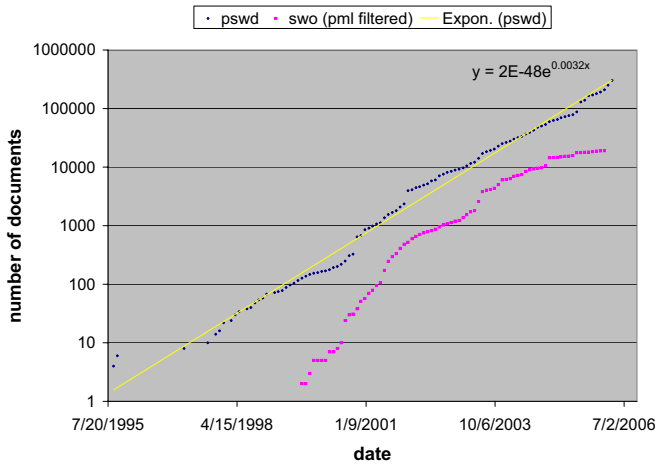


Figure V.6: Cumulative distribution of the last-modified date of PSWDs and SWOs

the birth of new version. Removing an SWD from the Web leads to the permanent death of the last version of the SWD. By comparing the distribution of last-modified time collected by Aug 2005 and Mar 2006 by Figure V.7, we have the following observations: (i) 163,566 PSWDs last-modified by August 2005 according to "05-AUG" have disappeared according to "06-MAR" because of being updated (70%) and being put offline (30%); and (ii) both curves finally achieve similar amount of PSWDs, and this observation indicates a balance of death rate and birth rate of static PSWDs[8].

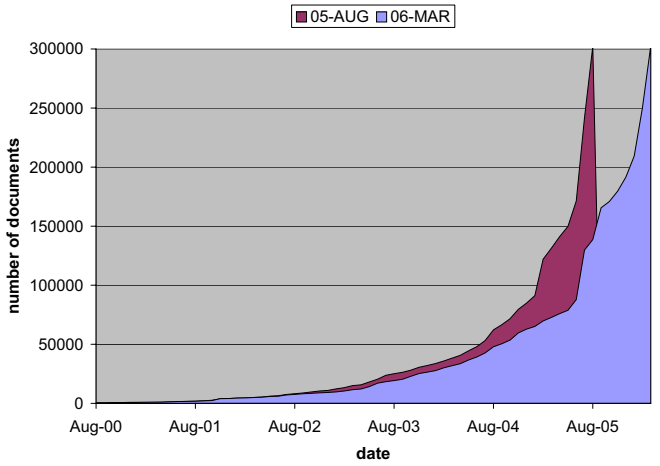


Figure V.7: The distribution of the number of PSWDs last modified before month t.

[8]Most PSWDs with last modified time are in fact static PSWDs

V.B.5 Size Change of SWD

In order to track the size change of SWDs, we maintain snapshots of each SWD once a new version has been detected. We use alive PSWDs to study **delta**, i.e., the change of size in terms of triples, and have the following observations:

- Among the 183,464 alive PSWDs that have at least three versions, 37,012 PSWDs have their sizes deceased since they have been initially discovered and they have lost a total of 1,774,161 triples; 73,964 PSWDs have their sizes increased and they have gained a total 6,064,218 triples, and the sizes of the rest 72,488 PSWDs do not change.
- Among those SWDs whose sizes remain unchanged, most are RSS files each of which changes its text content without altering the graph structure or the number of triples.
- Even though many SWDs have their sizes decreased, the amount of triples keeps increasing; hence, we may hypothesize that the size of the Semantic Web (in terms of the size of online triple space) is increasing.

Figure V.8 plots distribution of SWDs by the *delta*. Each unit in x-axis corresponds to a continuous range of delta, e.g. the span between 0 and 1 in x-axis corresponds to the delta value from 1 to 10. The "#triples" curve refers to the sum of delta (in terms of triples) for the certain delta value, and the "#swds" curve refers to the sum of SWDs. A peak in the curve is usually caused by the change of SWD generators.

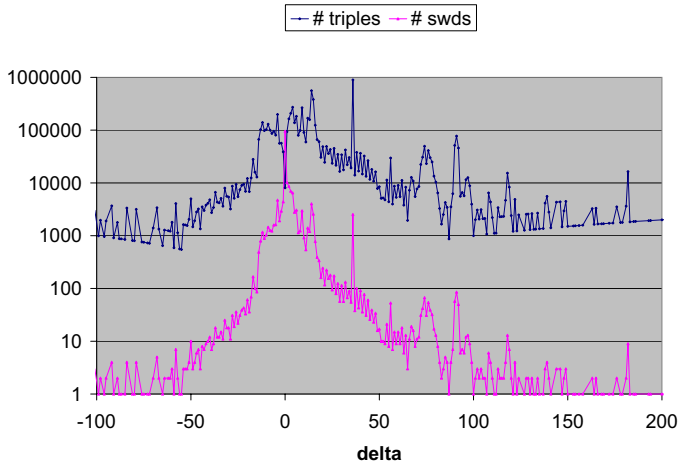


Figure V.8: The trend of size change of SWDs.

V.B.6 Definition Quality of SWD

In order to evaluate the portion of the definition in an SWD, we may compute *ontology ratio* (OntoRatio) at class-instance level and triple level. High *OntoRatio* implies a preference for adding term definition rather than populating existing terms; hence, *OntoRatio* can be used to quantify the degree of a Semantic Web document being a "real" ontology.

Given an RDF graph G, its class-instance level *OntoRatio* is computed using Equation V.1, where $C(G)$ and $P(G)$ refer to the set of SWTs being defined as classes and properties in G respectively, and $I(G)$ refers to the set of unique individuals (i.e., non-SWTs that instantiate SWTs as class). For example, given an RDF graph defining a class "Color" and instantiating the class with three class-instances, namely "blue", "green" and "red", the graph's *OntoRatio* is 0.25 because only one out of the four is defined as class. Therefore, we can conclude that the graph is intended to be a dataset instead of an ontology.

$$OntoRatio_{instance}(G) = \frac{|C(G)| + |P(G)|}{|C(G)| + |P(G)| + |I(G)|} \tag{V.1}$$

Class-instance level *OntoRatio*, however, could be biased in some cases. For example, when the class "Color" has been defined by 97 triples while the other three class-instances only occur in three triples, the graph should better be considered as an ontology. Therefore, we cannot not use instance level *OntoRatio*.

In this dissertation, We use triple level *OntoRatio* which is computed using Equation V.2, where $T(G)$ is the number of triples in G and $T_{definition}(G)$ is the number of definitional triples. *Definitional triples* include (i) triples that contain definition or reference of SWTs, (ii) triples that are related to the instance of *owl:Ontology*, (iii) triples that are connected to definitional triples by *blank nodes* within the RDF graph.

$$OntoRatio_{triple}(G) = \frac{T_{definition}(G)}{T(G)} \tag{V.2}$$

Figure V.9 plots the distribution of triple level *OntoRatio*: "swo" curve includes all SWOs that contain as least one *definitional triples*, "swo(onto.stanford.edu removed)" curve removes all SWOs from `onto.stanford.edu` as noises because they are PML documents and not intended to be ontologies.

The bias imposed by *onto.stanford.edu* leads to the study of triple level *OntoRatio* per website. We compute the average *OntoRatio* for each of the 761 websites hosting SWOs. Table V.4 lists top ten source websites by the number of unique SWOs (we have removed the duplicates), and we have the following observations:

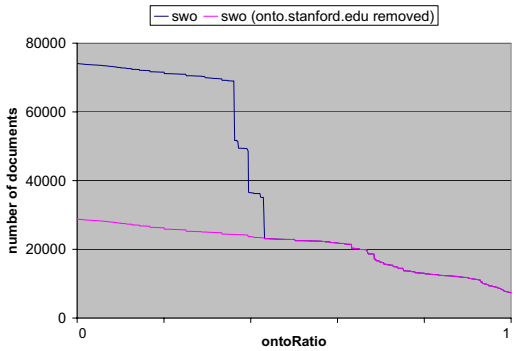


Figure V.9: The distribution of SWDs sharing the same triple level OntoRatio.

- SWOs from the first four websites have medium average *OntoRatio* but most of them are PML documents.
- Many SWOs from the fifth one `http://www.aifb.uni-karlsruhe.de/` are intended to store instance data of publications; however, an instance of *owl:Ontology* has been populated to specify the ontology for understanding the instance data.
- All SWOs from `xmlns.org` are intended to be ontologies.
- SWOs from the 7th, 8th and 9th websites have high average and high standard deviation because they are well known repositories of both SWOs and SWDS and they do host quite a few SWOs.
- The 10th website `http://lojjic.net/` has high *OntoRatio* but most SWDs from it are intended to be instance data storing Blog data.

Table V.4: Top ten largest source websites of SWOs

rank	site	#swo	avg(OntoRatio)	std(OntoRatio)
1	*http://onto.stanford.edu:8080/	45,278	0.39	0.03
2	*http://iw.stanford.edu/	3,215	0.67	0.06
3	*http://www.ksl.stanford.edu/	2,772	0.83	0.13
4	*http://iw4.stanford.edu/	2,040	0.60	0.13
5	*http://www.aifb.uni-karlsruhe.de/	1,822	0.19	0.12
6	http://xmlns.com/	1,161	1.00	0.00
7	http://www.mindswap.org/	912	0.68	0.31
6	http://www.w3.org/	907	0.82	0.26
9	http://cvs.sourceforge.net/	575	0.87	0.24
10	*http://lojjic.net/	524	0.74	0.18

* Most SWOs from the website are intended to be instance data.

Ontology Classification

An interesting question to Semantic Web researchers is how to tell if an SWD is intended to be an ontology.

We may use *OntoRatio* for this goal: an SWD is called SWO if its *OntoRatio* is greater than zero. However, such SWOs are not necessarily intended to be ontologies. For example, all PML documents intend to store instances of proof but they also contain redundant class and property definitions. Similar cases are *Semantic Blogging* pages[9], and *Semantic Bibliography* documents[10].

We may use the presence of class-instances of *owl:Ontology* for this goal. However, each Semantic Bibliography document includes an instance of *owl:Ontology* without the intension to be an ontology. This observation partially shows the difficulties in using ontologies: publisher cannot refer the definition of an SWT to SWOs other than the *official ontology* in Semantic Web dataset where *owl:imports* should not be used. Moreover, only 11,786 SWOs among the 74,120 SWOs do populate the instance of *owl:Ontology*.

A third approach uses a threshold α to find an approximate answer to this question. We empirically derive α as the mean of the website-wise average value of the triple-level *OntoRaio*. Intuitively, SWOs published at one website should have similar intentions and thus similar *OntoRatio* (our statistics shows that the standard deviation of *OntoRatio* from any of the 761 websites ranges between 0 and 0.5 and has a median of 0.005). Hence we use the average of the website-wise *OntoRatio* to derive $\alpha = 0.81$. Based on this criterion, many SWOs that intended to be instance datasets can be filtered, and we finally get 12,923 SWOs that intend to be ontologies.

V.C Measuring Semantic Web Vocabulary

A total of 237,645,189 triples have been accumulated in *SW06MAR* (we simply sum up triples from each SWDs without merging equivalent ones), including 1,415,054 distinct Semantic Web terms that use 11,648 Semantic Web namespaces.

V.C.1 Overall Usage of SWT

Table V.5 analyzes the usage of SWTs in SWDs base on the combination of the six types of meta-usage identified by the WOB ontology, namely, *hasClassDefinitionIn*, *hasPropertyDefinitionIn*, *hasClassInstanceIn*,

[9] e.g., `http://lojjic.net/blog/20020415-191800.rdf`
[10] e.g., `http://www.aifb.uni-karlsruhe.de/Publikationen/viewPublikationOWL/id1170.owl`

hasPropertyInsanceIn, *hasClassReferenceIn*, and *hasPropertyReferenceIn*. We have the following interesting observations:

- Only a few classes (1.53% – pattern 4) and properties (0.99% –pattern 7) are defined, referenced and populated. Some classes and properties are only populated without concrete definition (0.22% – patterns 11 and 12).
- Most SWTs (95.30% – patterns 1,2,3,6,8 and 10) are defined or referenced without being actually populated, while some SWTs (1.78% – patterns 5 and 9) are populated without being defined or referenced.
- Some SWTs (0.09% – pattern 15) mistakenly have both class and property meta-usage.
- Some SWTs (5.24% – patterns 3 and 10) are referenced without explicit class or property definition: a few are from *XMLSchma* that does not have RDF definition, and the rest are caused by various reasons, such as typo, ignorance on the definition of the referenced term, and inaccessible definition. For example, an N3 file at `http://simile.mit.edu/2003/10/ontologies/vraCore3` mistakenly used *dc:creator* as the super class of *vra:Entity*.

Table V.5: The usage patterns of Semantic Web terms

id	**SWT usage pattern**	**#swd**	%
1	defined as class but not populated	1,001,571	80.75
2	defined as property but not populated	91,238	7.36
3	referenced as class only	59,289	4.78
4	defined and populated as class	19,000	1.53
5	populated property without definition	14,266	1.15
6	defined as class with no more description or instances	12,929	1.04
7	defined and populated as property	12,326	0.99
8	defined as property with no more description or instances	11,291	0.91
9	popualted class without definition	7,761	0.63
10	referenced as property only	5,672	0.46
11	defined and populated as property without being referenced	1,940	0.16
12	defined and populated as class without being referenced	711	0.06
13	property usage without explicit definition	667	0.05
14	class usage without explicit definition	449	0.04
15	mistakenly used/defined/referenced as both class and property	1159	0.09

V.C.2 Definition Quality of SWT

The definition of an SWT depends on its residential RDF graph that is serialized by an SWD. Again, we count the number of definitional triples of the SWT to estimate the quality of its definition within an SWD. Usually,

important classes and properties have more definitional triples. Figure V.10 shows that the cumulative distribution of the quality of SWTs (i.e., the number of definitional triples) follows Power distribution. The imperfectness on the head and the tail of the curve reflect the convention of defining SWTs: we usually define SWT using a manageable number of triples (two to ten triples in most cases). From our statistics, the largest SWT definition could have as large as 1,000 triples: the SWT `http://127.0.0.1:8080/ontologies/DomainOntologies/middle_ontology#MOSemanticRelationType` has 973 definitional triples in the SWD `http://elikonas.ced.tuc.gr/ontologies/DomainOntologies/middle_ontology`. The "relation" and "annotation" curves are generated by splitting the definitional triples into two groups (i.e., annotation triples whose *rdf:object* is *rdfs:Literal* and relation triples whose *rdf:object* is *rdfs:Resource*), and then counting their distributions separately. From the figure, we observe strong preferences on relation triples because ontology engineers are too tired to provide natural language descriptions and the local-name of an SWT is usually assumed self-descriptive.

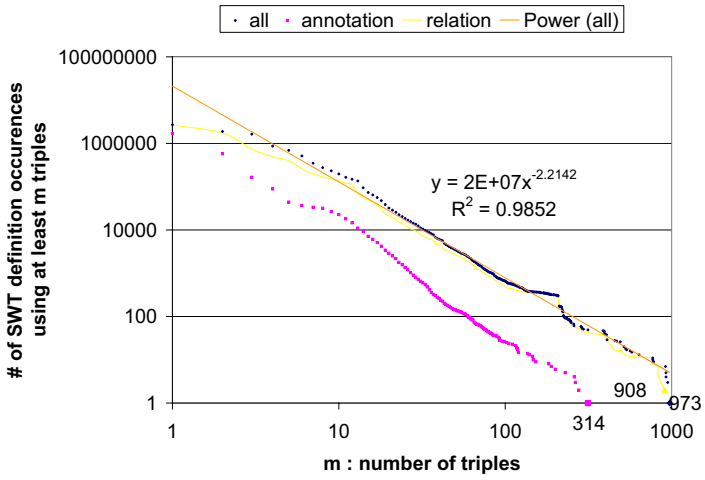


Figure V.10: The cumulative distribution of definition quality of SWT

With the global view of the Semantic Web, we may further evaluate SWT definitions obtained from multiple sources based on the intuition that the quality of a definitional triple depends on the number of sources supporting it. Here, we only show the significance of this problem and will discuss how to aggregate the definitions of an SWT from multiple sources in Chapter VII. We have found 104,152 SWTs being defined by more than one SWDs.

V.C.3 Instance Space of SWT

A common question posed by Semantic Web knowledge consumers is what kind of Semantic Web data is available. We answer this question by measuring the instance space of the Semantic Web, i.e., how SWTs are populated in SWDs as classes and properties.

Figure V.11 plots the cumulative distribution of the number of SWTs being populated as class (or property) by at least m instances (or SWDs). All the four curves follow Power distribution again. Only a few SWTs have been well populated: only 371 SWTs have been instantiated as classes by more than 100 SWDs; only 1,706 SWTs have more than 100 class-instances; only 1,208 SWTs have been instantiated as properties by more than 100 SWDs; and about 4,642 SWTs have been instantiated as properties for more than 100 times.

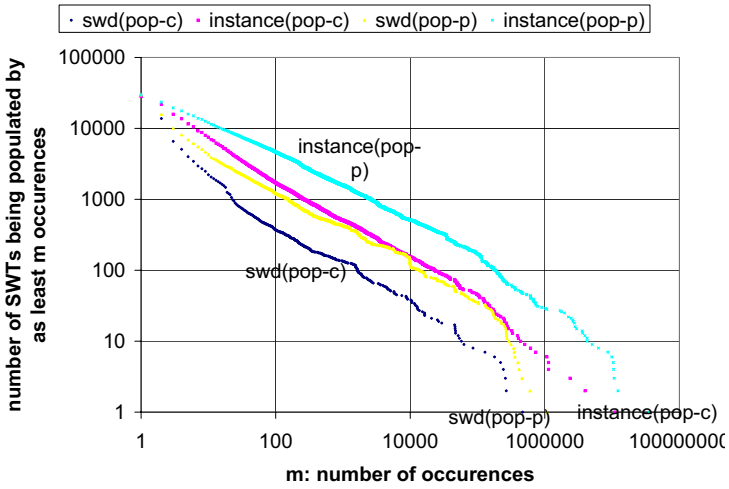


Figure V.11: The distribution of instance space of SWT

Table V.6 lists popular SWTs being populated as class ordered by the number of source SWDs and the number of instances respectively. Usually, the number of an SWT's class-instances is proportional to the number of SWDs populating the SWT; however, exceptions exist. For example, `http://www.cogsci.princeton.edu/~wn/schema/Noun` has significant number of class-instances but is populated by only a few giant SWDs.

Table V.7 lists popular SWTs being instantiated as properties ordered by the number of source SWDs and the number of instances respectively. Our observations show that ontologies (RDF, RDFS, OWL), library alike metadata (DC), social network profile (FOAF), and online news feeds (RSS) constitute the dominate portion of the current Semantic Web data space.

Table V.6: SWTs that are most instantiated as classes

resource URI	#swd	#instance
Most instantiated classes ordered by #swd		
http://xmlns.com/foaf/0.1/Person	467,806	11,040,981
http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq	267,603	277,608
http://purl.org/rss/1.0/channel	259,417	265,700
http://purl.org/rss/1.0/item	241,984	3,971,918
http://xmlns.com/foaf/0.1/Document	220,064	242,994
http://xmlns.com/foaf/0.1/PersonalProfileDocument	178,946	178,975
http://www.w3.org/2003/01/geo/wgs84_pos#Point	85,695	107,859
http://www.w3.org/2002/07/owl#Class	62,867	1,075,220
http://www.w3.org/1999/02/22-rdf-syntax-ns#Property	57,561	503,829
http://www.w3.org/1999/02/22-rdf-syntax-ns#List	53,726	54,491
Most instantiated classes ordered by #instance		
http://xmlns.com/foaf/0.1/Person	467,806	11,040,981
http://purl.org/rss/1.0/item	241,984	3,971,918
http://www.cogsci.princeton.edu/~wn/schema/Noun	36	2,376,900
http://xmlns.com/wordnet/1.6/Person	2,823	1,138,374
http://xmlns.com/foaf/0.1/chatEvent	2,693	1,138,182
http://www.w3.org/2002/07/owl#Class	62,867	1,075,220
http://www.nlm.nih.gov/mesh/2004#Concept	18	734,706
http://www.daml.org/2002/02/telephone/1/areacodes-ont#Exchange	768	614,400
http://www.w3.org/1999/02/22-rdf-syntax-ns#Property	57,561	503,829
http://www.cogsci.princeton.edu/~wn/schema/Verb	36	436,572

Table V.7: SWTs that are most instantiated as properties

resource URI	#swd	#instance
Most instantiated properties ordered by #swd		
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	1,075,644	35,391,332
http://purl.org/dc/elements/1.1/title	600,902	12,220,239
http://xmlns.com/foaf/0.1/mbox_sha1sum	456,879	2,608,134
http://purl.org/dc/elements/1.1/description	414,120	2,463,056
http://www.w3.org/2000/01/rdf-schema#seeAlso	398,162	10,754,486
http://xmlns.com/foaf/0.1/nick	359,692	9,894,684
http://xmlns.com/foaf/0.1/knows	350,312	10,439,805
http://xmlns.com/foaf/0.1/weblog	345,362	8,623,152
http://webns.net/mvcb/generatorAgent	314,099	336,904
http://purl.org/dc/elements/1.1/date	292,757	5,076,308
Most instantiated properties ordered by #instance		
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	1,075,644	35,391,332
http://purl.org/dc/elements/1.1/title	600,902	12,220,239
http://www.w3.org/2000/01/rdf-schema#seeAlso	398,162	10,754,486
http://xmlns.com/foaf/0.1/interest	164,234	10,529,470
http://xmlns.com/foaf/0.1/knows	350,312	10,439,805
http://xmlns.com/foaf/0.1/nick	359,692	9,894,684
http://xmlns.com/foaf/0.1/weblog	345,362	8,623,152
http://www.cogsci.princeton.edu/~wn/schema/wordForm	36	6,264,072
http://purl.org/dc/elements/1.1/date	292,757	5,076,308
http://purl.org/rss/1.0/link	268,661	4,264,349

V.C.4 Instantiation of *rdfs:domain*

Since Semantic Web data is published asynchronously, the publishers may populate instances using multiple existing ontologies. We can reverse-engineer the definitions in ontologies by learning the instances of ontologies. In particular, we focus on the instantiation of *rdfs:domain* relation which associates a class with properties for describing its instance data. So far we have observed 91,707 unique instantiations of *rdfs:domain*.

Figure V.12 plots the distribution of the number of instantiations having been observed in at least m instances ("instance" curve) and SWDs ("swd" curve) respectively. Again, Power distribution has been observed.

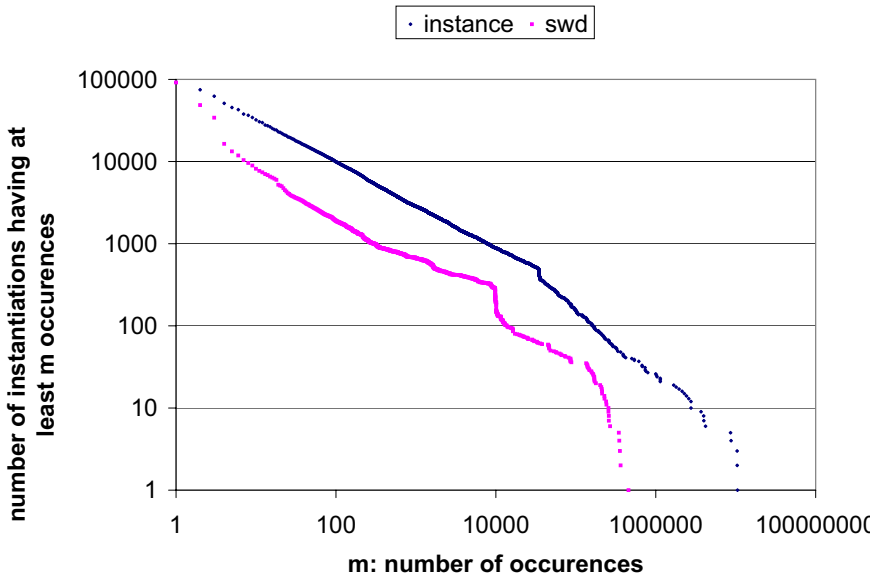


Figure V.12: The cumulative distribution of instantiations of *rdfs:domain*

Table V.8 lists top ten most popular instantiations of *rdfs:domain* ordered by the number of SWDs and instances respectively. The highly instantiated *rdfs:domain* relations are mainly from popular instance space such as FOAF documents and RSS documents. An interesting observation is that *rdfs:seeAlso* has been frequently used as *instance property* of *foaf:Person*. This practice cannot be found in RDFS ontology or FOAF ontology although it has been informally described in FOAF specification [20]. We also noticed that giant Semantic Web dataset such as WordNet dataset dump[11].

[11] `http://www.cogsci.princeton.edu/~wn/schema/`

Table V.8: Popular instantiations of *rdfs:domain*

class URI/property URI	#swd	#instance
Most instantiated *rdfs:domain* definition ordered by #swd		
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/mbox_sha1sum	456,649	2,586,502
http://xmlns.com/foaf/0.1/Person http://www.w3.org/2000/01/rdf-schema#seeAlso	363,688	10,400,974
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/nick	356,209	8,741,895
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/knows	350,276	10,440,827
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/weblog	345,106	8,620,885
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/homepage	268,272	1,675,421
http://purl.org/rss/1.0/channel http://purl.org/rss/1.0/title	259,244	262,164
http://purl.org/rss/1.0/channel http://purl.org/rss/1.0/link	259,086	259,935
http://purl.org/rss/1.0/channel http://purl.org/rss/1.0/description	256,450	258,781
http://purl.org/rss/1.0/channel http://purl.org/rss/1.0/items	256,448	256,937
Most instantiated *rdfs:domain* definition ordered by #instance		
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/interest	164,179	10,528,365
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/knows	350,276	10,440,827
http://xmlns.com/foaf/0.1/Person http://www.w3.org/2000/01/rdf-schema#seeAlso	363,688	10,400,974
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/nick	356,209	8,741,895
http://xmlns.com/foaf/0.1/Person http://xmlns.com/foaf/0.1/weblog	345,106	8,620,885
http://www.cogsci.princeton.edu/~wn/schema/Noun http://www.cogsci.princeton.edu/~wn/schema/wordForm	36	4,189,032
http://purl.org/rss/1.0/item http://purl.org/rss/1.0/link	241,833	3,990,004
http://purl.org/rss/1.0/item http://purl.org/rss/1.0/title	239,629	3,977,503
http://purl.org/rss/1.0/item http://purl.org/rss/1.0/description	227,422	3,684,670
http://purl.org/rss/1.0/item http://purl.org/dc/elements/1.1/date	208,363	2,756,315

Based on the observations on the instantiation of *rdfs:domain*, we may easily find the most used properties of a given class, and then either modify ontology to reflect the missing domain definition or directly populate more class instances following such well-adopted convention. Table V.9 lists the best three properties for some popular classes as long as the corresponding *rdfs:domain* relations have been instantiated by over 10,000 SWDs.

Table V.9: Popular properties of a popular class

class URI	property URI	#swd
rss:item	rss:link	241,833
	rss:title	239,629
	rss:description	227,422
rss:channel	rss:title	259,244
	rss:link	259,086
	rss:description	256,450
foaf:Person	foaf:mbox_sha1sum	456,649
	rdfs:seeAlso	363,688
	foaf:nick	356,209
foaf:PersonalProfileDocument	foaf:maker	178,530
	foaf:primaryTopic	170,497
	mcvb:generatorAgent	170,390
rss:image	rss:url	10,508
	rss:title	10,501
	rss:link	10,494
foaf:Document	dc:title	213,374
	dc:description	213,005
geo:Point	geo:lat	85,694
	geo:long	85,693
foaf:Image	foaf:thumbnail	18,444
	foaf:page	15,611
	dc:description	15,317
cc:License	cc:permits	24,303
	cc:requires	22,779
	cc:prohibits	12,960
foaf:OnlineAccount	foaf:accountServiceHomepage	25,389
	foaf:accountName	25,381
foaf:OnlineChatAccount	foaf:accountName	10,526
http://www.daml.org/2001/10/html/airport-ont#Airport	geo:long	12,448
	geo:lat	12,448
http://www.hackcraft.net/bookrdf/vocab/0_1/Publisher	foaf:name	16,369
	dc:title	13,678
http://purl.org/dc/dcmitype/Image	dcterms:modified	10,185
	dcterms:extent	10,143
	dc:title	10,120
cc:Work	cc:license	21,673
dcterms:W3CDTF	rdf:value	12,766
	rdfs:label	12,213
dcterms:RFC1766	rdf:value	12,774
	rdfs:label	12,057
dcterms:IMT	rdf:value	13,016
	rdfs:label	12,305
http://purl.org/dc/dcmitype/Text	dcterms:modified	11,550
	dcterms:created	11,550
	dc:identifier	11,549
iw:NodeSet	iw:isConsequentOf	46,496
	iw:hasLanguage	46,496
	iw:hasConclusion	32,559

V.D Navigation Quality

The Web infrastructure grants the Semantic Web the power of distributing knowledge across the Web, however, enough navigational paths are needed to facilitate consumers to access the distributed Semantic Web data. This section analyzes three types of existing navigational paths in the Semantic Web:

- The triples instantiating *owl:imports* enable paths between SWOs.
- The namespace of a resource enables a path from an SWD to an SWO.
- The triples instantiating *link-indicators* (such as *rdfs:isDefinedBy*) enable paths between SWDs.

V.D.1 Paths based on Imports Relations

Table V.10 shows the performance of *imports* statements in *SW06MAR*: "#total" refers to the number of triples that instantiate the property, and "#swd" refers to the number of triples that instantiate the property and having SWDs as the *rdf:object*. Here, we count *imports* statements by the presence of *owl:imports* or other properties having local-name "imports". We also notice that *imports* statement may link to a Web document or an unreachable URL because the referenced SWDs are maintained separately. In the table, *owl:imports* and *daml:imports* are the most used. Unfortunately, the total number of *imports* statements is less than 10,000 and only 5,442 SWDs (out of the 1 million SWDs) have been linked by these relations. Hence, we cannot rely on *imports* statement to navigate the Semantic Web.

Table V.10: Performance of SWTs that indicates *imports* statement

"imports" property URI	#total	#swd
http://www.w3.org/2002/07/owl#imports	7,041	6,701
http://www.daml.org/2001/03/daml+oil#imports	1,731	1,474
http://www.daml.org/2000/10/daml-ont#imports	187	185
http://www.daml.org/2000/12/daml+oil#imports	32	29
http://www.daml.org/2001/03/daml+oil.daml#imports	27	24
http://www.w3.org/2003/07/ferrell#imports	10	0
http://www.w3.org/2001/10/daml+oil#imports	10	10
Http://www.w3.org/2002/07/owl#imports	10	10
http://www.daml.org/2001/03/daml-ont#imports	7	7
http://www.daml.org/2001/03/daml-oil#imports	3	3
http://www.cs.man.ac.uk/~horrocks/daml+oil/datatypes/daml+oil+dt#imports	2	1
http://www.w3.org/2000/10/swap/test/ferrell/ferrell#imports	2	0
http://www.daml.org/2000/11/daml-ont#imports	2	2
http://www.w3.org/2000/08/daml-ont#imports	1	0
http://www.daml.org/2000/11/daml-oil#imports	1	1

V.D.2 Paths based on Term's Namespace

Due to the limited number of *imports* paths between SWOs, we may need to use an SWT's namespace. Table V.11 lists top ten protocols use by SWTs and corresponding namespaces. "http" is the dominating one because it supports Web addressing; however, the rest protocols are seldom used because they seldom support Web addressing, e.g., "tag" protocol is basically localized even though it has quite a few population. "htttp" is an obvious typo, and "foo" is usually used for creating examples.

Table V.11: Top ten protocols used by SWTs

protocol	#SWTs	#NSs	example namespace URI
http	1,403,367	8,899	`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
file	5,807	235	`file:/home/em/w3ccvs/WWW/2001/07/25-swws/#_gs2`
urn	5,454	49	`urn:bornstein:item-statusmailed`
meh	510	3	`meh://880762352#square`
htttp	398	2	`htttp://augmented.man.ac.uk/ontologies/TravelOntology.owl#PathBreadcrumb`
tag	193	19	`tag:decoy@iki.fi,2000:rdf:associations:1.0:permitsRoles`
webode	92	6	`webode://knowledgeweb.semanticweb.org/Project+Ontology#Task`
https	66	7	`https://launchpad.net/rdf/launchpad#Product`
voc	60	9	`voc://plx.abc.com/foo/ModernDate`
foo	46	5	`foo://graph-layout.n3#same`

We focus on the SWNs using HTTP protocol, and find their **official ontology**, i.e., an SWO that provide term definitions for a given SWN. Theoretically, the URL of an SWN should always link to an official ontology; however, only half of the SWNs in *SW06MAR* have successfully link to valid SWOs. Therefore, we employee the following heuristics to find *official ontology* of a given Semantic Web term t:

1. The SWO whose URL is the same as the namespace of t
2. The SWO whose URL is redirected from the namespace of t. For example, `http://purl.org/dc/elements/1.1/` is redirected to `http://dublincore.org/2003/03/24/dces`).
3. The SWO whose URL is the concatenation of t's namespace and "index.rdf" (or "schema.rdf"). For example, `http://xmlns.com/foaf/0.1/index.rdf` is the official ontology of `http://xmlns.com/foaf/0.1/`).
4. The SWO is the only SWO whose URL has t's namespace as prefix.

Although heuristics 2, 3 and 4 has so far contributed only 16 additional *official ontology* relations, the path from FOAF namespace to FOAF ontology is found by them.

V.D.3 Paths based on Link Indicators

Beside paths to SWOs introduced by *imports* statement and official ontology, there are still many paths linking SWDs via *link indicators*, i.e., properties highly possible linking the present SWD to another SWD. Table V.12 lists top ten link indicators out of 138 confirmed ones. Note that such link indicators require that the subject of corresponding triple be the same as the URL of the residential SWD. In the table, *rdfs:seeAlso*'s link semantics is mainly enforced in FOAF documents, the second one *cc:license*, and the third one *dc:type* are enforced by certain software tools and link to one or several common SWDs. With these link indicators, we have collected 103,249 links connecting 224,234 SWDs.

Table V.12: Top link indicators (excluding imports)

property URI	#total	#valid
`http://www.w3.org/2000/01/rdf-schema#seeAlso`	66,266	60,373
`cc:license`	29,471	19,410
`http://purl.org/dc/elements/1.1/type`	4,154	4,003
`http://swrc.ontoware.org/ontology#publication`	3,552	3,449
`http://swrc.ontoware.org/ontology#author`	3,305	3,305
`http://purl.org/dc/elements/1.1/source`	7,715	1,860
`http://swrc.ontoware.org/ontology#isAbout`	1,855	1,855
`http://web.resource.org/cc/#License`	967	967
`http://swrc.ontoware.org/ontology#projectInfo`	776	768

V.E Summary

In this chapter, we justify the significance of our sample dataset *SW06MAR* in approximating the current Semantic Web on the Web. Additionally, we use the dataset, together with Google meta-search, to bound the size of the current Semantic Web. The growth of the Semantic Web has been positively supported by the exponential distribution of SWD's age and the overall growth of SWD's size.

We measure the deployment status of the Semantic Web on the Web with respect to the Web and the RDF graph world. In particular, a series of quantitative metrics and in-depth analysis bring a global picture of the SWDs and SWTs in the Semantic Web. (Invariant) Power distribution has been observed in many cases, such as the distribution of SWDs per website and the definition quality of SWT. We also notice that the bias introduced by the dynamic SWDs could block the diversity of the Semantic Web and should be controlled.

The navigational paths in the Semantic Web are still in small amount and not enough for effective Semantic Web surfing. We will discuss our solution to this issue in next chapter.

Chapter VI

SURFING THE SEMANTIC WEB

While the Web makes it easy to publish Semantic Web data, accessing online Semantic Web data is still difficult to information consumers because of the open architecture of the Web: users need the Web address of Semantic Web data before accessing it. Unfortunately, neither conventional Web search nor conventional Web surfing can effectively address this issue because Semantic Web data is sparsely distributed on the Web.

In this chapter, we focus on the *accessibility* issue in Web-scale Semantic Web data access, i.e., how to help the consumers to retrieve the desired Semantic Web data on the Web. To this end, we build a conceptual model to capture the unique behaviors that distinguish Semantic Web surfing from conventional Web surfing, and then implement a Semantic Web search engine – Swoogle to realize the model. Finally, we use the conceptual model to design an explainable ranking scheme that orders Semantic Web data by popularity.

VI.A Semantic Web Search and Navigation Model

VI.A.1 Entities Accessed Semantic Web Surfing

While Web documents are the only entities in Web surfing, Semantic Web surfing involves many accessible entities due to the structural organization of Semantic Web data. In the WOB ontology, we have identified several important entities that are frequently accessed in Semantic Web surfing as the following:

- *Semantic Web documents* are the data transfer packets in Semantic Web data access, i.e., consumers retrieve online Semantic Web data by downloading Semantic Web documents from the Web.

- *Semantic Web ontologies* are special Semantic Web documents that define a group of related Semantic Web terms. They can be used to restrict search space and highlight additional navigational paths.

- *Semantic Web terms* function like words in natural language, i.e., consumers associate the concepts in their mind to Semantic Web terms in the RDF graph world. It is notable that users access SWTs with their meta-usage, i.e. finding all definitions of an SWT on the Semantic Web.

- *Semantic Web namespaces* are special RDF resources that are used as namespaces by Semantic Web terms or Semantic Web documents. A Semantic Web namespace is a useful node in Semantic Web surfing, for example, users may surf from an SWT to its namespace and then to the *official ontology*.

Entities from the agent world are excluded for two reasons: (i) they are not critical in addressing and retrieving Semantic Web data on the Web although they are important in accepting or rejecting the fetched data; and (ii) there are not enough provenance relations from the entities in the RDF graph world to the agent world even though the number of "person" entities is huge.

RDF graph reference is excluded for two reasons: (i) users usually care the entire RDF graph serialized by an SWD but not the sub-graphs; and (ii) accessing RDF graph usually requires huge storage space that stores all RDF graphs in full and efficient data access mechanisms.

VI.A.2 Unique Behaviors in Semantic Web Surfing

Figure VI.1 illustrates a typical Web-scale Semantic Web data access process.

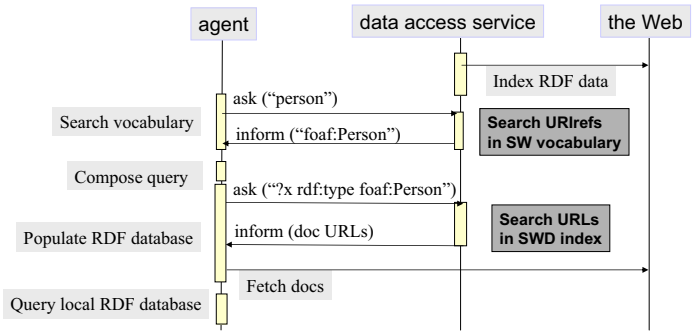


Figure VI.1: A typical Web-scale Semantic Web data access process

The process involves an agent, a data access service, and the Web: (i) the data access service harvests the Semantic Web and builds an index of online Semantic Web data; (ii) the agent asks the data access service for SWTs using keyword query "person" and is informed an SWT *foaf:Person*; (iii) the agent composes a query using *foaf:Person* and Semantic Web query languages; (iv) the agent asks the data access service to search relevant SWDs using the query and composes a local RDF database by fetching the informed SWDs; and (v) the agent queries the local RDF database using the query.

The uniqueness of surfing behaviors in accessing Semantic Web data is mainly caused by the semantic structure and content of Semantic Web data. We summarize three types of basic behaviors in Web-scale Semantic Web data access:

- **search.** A Semantic Web document can be treated as a conventional Web document. Therefore, users may want to match SWDs by keyword queries written in natural languages [127, 164]. For example, searching all "person" ontologies using keyword *person*. Similarly, the URI of Semantic Web term can be treated as literal string during search. The advances in *semantic annotation systems* can enrich the search interface with more complex search support: users may specify constraints in (property, value) style on the structured meta-description of Web documents.

- **query.** We may also query the RDF graph of a Semantic Web document. In this case, the query is written in Semantic Web query languages [71, 133], and the query results can be URLs of SWDs or RDF graphs depending on the query language. For example, finding all SWTs that are *rdf:subClassOf foaf:Agent*. While query is normally thought of as an important operation in RDF database systems, we may extend it to Web-scale Semantic Web data access, where a *query* is executed in two steps: (i) to retrieve all relevant SWDs to form a unified RDF graph and (ii) to run query on the RDF graph.

- **navigation.** Users may also use special *queries* to utilize the navigation network that interlinks the entities in the Semantic Web and achieve Semantic Web surfing. For example, enumerating all RDF nodes whose *rdf:type* is *foaf:Person*. In order to achieve navigation, we need to generate a navigation network that interlinks entities in advance and thus support navigation operations.

When accessing Semantic Web data, human users prefer the combination of search and navigation to avoid the use of Semantic Web query languages, but machine agents, prefer the combination of query and navigation to utilize Semantic Web query languages. In this dissertation, we focus on the combination of search and navigation and leave the other combinations to future work.

We build conceptual model to characterize Semantic Web surfing behaviors. The model consists of three major components:

- the accessible entities – we focus on the four types of entities in the Semantic Web (see Section VI.A.1)

- the meta-description of entities and corresponding search mechanisms

- the navigation network that interlinks the entities and corresponding navigation mechanisms.

VI.A.3 The Current Search and Navigation Model

Based on existing technologies, the current Semantic Web data access activities are modeled in Figure VI.2. The *search* component is implemented based on Conventional Web search engines, e.g., users use Google to search the desired Semantic Web documents. Therefore, Semantic Web documents are accessed as Web documents and the semantic content and structure in Semantic Web documents cannot be used to search SWDs. The *navigation* component is implemented based on parsing Semantic Web documents and Semantic Web terms. The navigation network has eight distinct types of paths as the following:

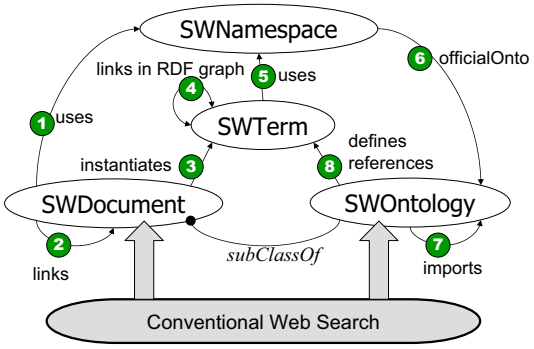


Figure VI.2: The current Semantic Web search and navigation model

Here, search interface is denoted by block arrow, navigational path is denoted by arrow, and the sub-class relation is denoted by round-end arrow. Paths 3 and 8 are straightforward according to meta-usage of SWTs in corresponding SWDs and SWOs. Paths 1, 2, 6, and 7 are supported by most existing RDF browsers after translating URIs to hyperlinks. Path 5 requires special treatment in extracting namespace from SWT. Path 4 is limited within the residential SWD.

path 1 *document-namespace relation.* The Semantic Web namespaces used by an SWD can be extracted from the namespace declaration part of the SWD as well as the namespace of a URI in the SWD. Hence, users may navigate to the used Semantic Web namespaces.

path 2 *document-document relation.* SWDs can be linked by *link-indicators* (see Section V.D.3), for example, *rdfs:seeAlso* and *rdfs:isDefinedBy* are often used to link the present SWD to another SWD. Hence, users may navigate to the linked SWDs.

path 3 *document-term relation.* Some SWTs are instantiated but not defined or referenced by the present SWD. Hence, users may navigate to an instantiated SWT.

path 4 *term-term relation.* Within the RDF graph of the present SWD, users may navigate from one SWT to another SWT.

path 5 *term-namespace relation.* From a SWT, users may navigate to its Semantic Web namespace obtained from URI parsing.

path 6 *namespace-ontology relation.* The semantics of Semantic Web namespace indicates that users may locate and navigate the official SWO using the URL of the namespace.

path 7 *ontology-ontology relation.* SWOs may be interconnected by sub-properties of *owl:OntologyProperties* such as *owl:imports*, *owl:priorVersion*, *owl:backwardCompatibleWith*, and *owl:incompatibleWith*. Hence, users may navigate from the present SWO to another SWO.

path 8 *ontology-term relation.* Besides instantiating SWTs, an SWO may have other meta-usage (i.e., define or reference) of SWTs. Hence, users may navigate from the present SWO to a defined SWT.

The above model is imperfect due to the limitations of existing technologies. First, it is hard to find Semantic Web documents using conventional search engines because they are designed to search Web documents and seldom index the semantic content and structure of Semantic Web documents. Second, it is hard for users to map their concepts to Semantic Web terms due to the lack of interactions between information providers and consumers. Third, it is hard to navigate the Semantic Web due to the low connectivity of the navigation network. For example, *rdfs:seeAlso* has been widely used to interconnect FOAF documents, but it seldom works in other SWDs; *owl:imports* does interlink Semantic Web ontologies, but it has been poorly instantiated and the SWOs contribute a tiny part of the Semantic Web. Last, some navigational paths, such as surfing back from a Semantic Web term to the Semantic Web documents that have the SWT's meta-usage, are very useful but absence because computing such paths requires the global catalog of the Semantic Web. In summary, the current search and navigation model need to be enhanced to promote Web-scale Semantic Web data access.

VI.A.4 The Enhanced Search and Navigation Model

Based on the WOB metadata, we enhance both *search* and *navigation* components of the current model as shown in Figure VI.3. In this figure, block arrows denote search interfaces, arrows denote navigational paths, and round-end arrow denotes the sub-class relation.

The *search* component is enhanced by a specialized Semantic Web search engine, which replaces the conventional web search engines. It only uses confirmed Semantic Web documents (including Semantic Web ontologies and embedded Semantic Web documents) to generate meta-description for SWDs, SWOs, SWTs

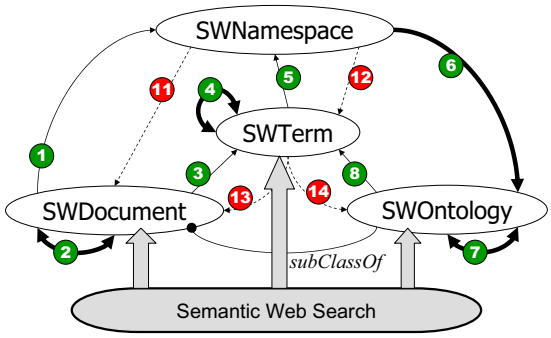


Figure VI.3: The enhanced Semantic Web search and navigation model

and SWNs. The meta-description of SWD and SWO is based on the parsing result of the semantic content and structure. For example, only the definition part of an SWO is covered by the meta-description. Moreover, the meta-description about an SWT is generated by aggregating the definitions from SWOs that define or reference the SWT.

The *navigation network* is greatly enriched by the WOB metadata that is derived from the global catalog of the Semantic Web. With the global catalog, we can enumerate all SWDs that instantiate *foaf:Person* as class. In particular, four types of navigational paths are added as the following:

path 11 *namespace-document relation.* Users may surf from a namespace to the Semantic Web documents using it. This relation is particularly useful in find Semantic Web documents. For example, we can search for all SWDs that use the Inference Web namespace[1].

path 12 *namespace-term relation.* Users may surf from a namespace to the Semantic Web terms using it.

path 13 *term-document relation.* Users may surf from a Semantic Web term to the Semantic Web documents instantiating but not defining or referencing it. One usage of such relation is to find class-instances of a particular Semantic Web term. In WOB ontology, two provenance properties contribute to this relation, namely *hasClassInstanceIn* and *hasPropertyInstanceIn*.

path 14 *term-ontology relation.* Users may surf from a Semantic Web term to the Semantic Web documents defining or referencing it. In the WOB ontology, four provenance properties contribute to this relation, namely *hasClassDefinitionIn*, *hasPropertyDefinitionIn*, *hasClassReferenceIn*, and *hasPropertyReferenceIn*.

[1]The current Inference Web namespace is `http://inferenceweb.stanford.edu/2004/07/`

Moreover, the new model also significantly enriches existing navigational paths as the following:

- *path 2 and 7* have been enriched by reverse links so that users may surf back to the Semantic Web documents linking to the present document.
- *path 4* has been enriched by global links aggregated from individual Semantic Web documents so that users may surf between Semantic Web terms without retrieving SWDs.
- *path 6* has been enriched by additional official ontology links heuristically computed from the global metadata about the Semantic Web. The details of deriving official ontology have been elaborated in Section V.D.2.

Figure VI.4 demonstrates how the enhanced model works through an example. We elaborate several interesting scenarios as the following:

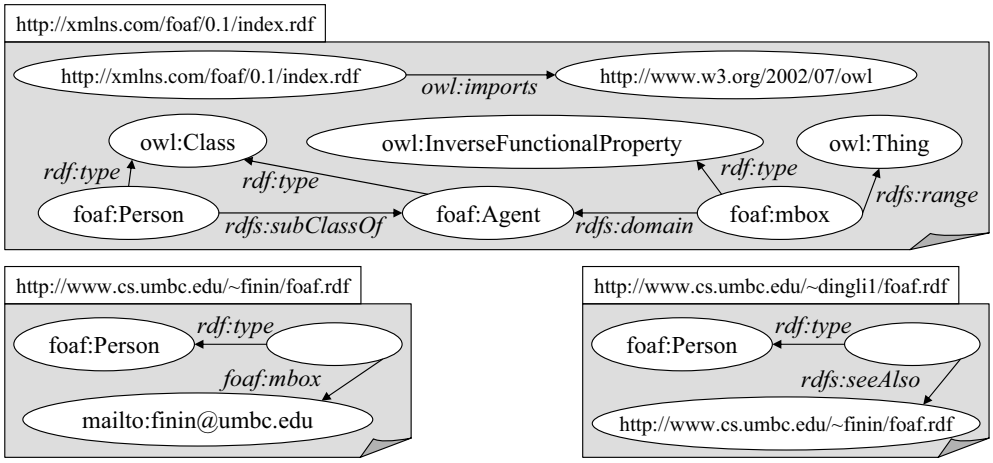


Figure VI.4: A use-case for the enhanced search and navigation model

- A user can use term search feature provided by Semantic Web search to find SWTs having local name "Person". Once she surfs to the SWT *foaf:Person*, she is then inside the Semantic Web.
- From the SWT *foaf:Person*, she can jump to the corresponding SWO `http://xmlns.com/foaf/0.1/index.rdf` by following *path 14* via *hasClassDefinitionIn* provenance property, to an SWD `http://www.cs.umbc.edu/~dingli1/foaf.rdf` by following *path 13* via *hasClassInstanceIn*, to another SWT *foaf:mbox* whose *rdfs:domain* is the present SWT by following *path 4*, or to an SWN `http://xmlns.com/foaf/0.1/` by following *path 5* via *hasNamespace*.

- From the SWO `http://xmlns.com/foaf/0.1/index.rdf`, she can jump to another SWO `http://www.w3.org/2002/07/owl` by following *path 7* via *owl:imports*, or to an SWT *rdfs:domain* by following *path 8* because the SWT has been referenced by the SWO.

- From the SWN `http://xmlns.com/foaf/0.1/`, she can jump to an SWO `http://xmlns.com/foaf/0.1/index.rdf` by following *path 6* via *hasOfficialOntology*, to an SWD `http://www.cs.umbc.edu/~finin/foaf.rdf` by following *path 11* via *hasNamespaceUsageIn*, or to an SWT *foaf:mbox* by following *path 12* because the SWN is used by the SWT as namespace.

- From the SWD `http://www.cs.umbc.edu/~dingli1/foaf.rdf`, she can jump to another SWD `http://www.cs.umbc.edu/~finin/foaf.rdf` by following *path 2* via *rdfs:seeAlso*, to an SWT *rdfs:seeAlso* by following *path 3* because the SWT has been instantiated by the SWD, or to an SWN `http://xmlns.com/foaf/0.1/` by following *path 1* be cause the SWN has been declared by the SWD.

VI.B Swoogle: A Semantic Web Search Engine

Swoogle [40] is a Semantic Web search engine that discovers, indexes, analyzes and searches Semantic Web documents and Semantic Web terms published on the Web. It aims at supporting the enhanced Semantic Web search and navigation model and thus facilitating Web-scale Semantic Web data access.

VI.B.1 Architecture

Like conventional web search engines, Swoogle is designed with the following high level tasks: discovering or revisiting online Semantic Web documents, indexing corresponding metadata to form a global view of the Semantic Web, and providing service interface to answer users' queries.

When implementing these tasks, two facts should be noticed. First, we are processing Semantic Web documents that are distinct from and in far less amount than regular Web documents. Second, the intended users of Swoogle are both software agents (the majority), who usually search SWDs for external knowledge and then retrieve SWOs to fully understand SWDs, and Semantic Web researchers (in non-trivial amount), who mainly search SWTs and SWOs for study or publishing instance data. Figure VI.5 depicts the architecture of Swoogle that includes four major components.

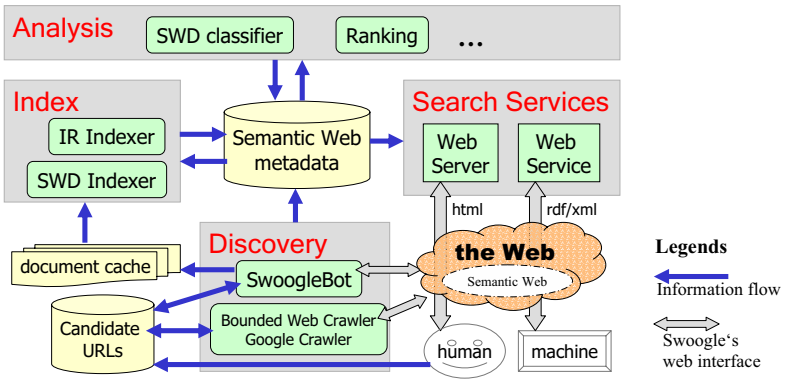


Figure VI.5: The architecture of Swoogle

- The **Discovery** component focuses on harvesting SWDs. It collects candidate URLs of SWDs using the harvesting methods mentioned in Chapter IV, including manual submission, Google based meta-crawling, bounded HTML crawling, and RDF crawling. For each candidate URL, it validates the downloaded content and tries to extract RDF graph. More over, it maintains snapshots for each newly discovered or updated SWDs to replicate the idea of the Internet Archive (`http://www.archive.org/`): tracking versions of Web documents.

- The **Indexing** component analyzes the properties of each discovered SWD and the corresponding parsed RDF graph, and then generates the bulk of Swoogle's metadata about the Semantic Web. The metadata not only characterizes the features that describes individual SWDs (or SWTs), but also tracks the relations between SWDs and SWTs, e.g., "how SWDs reference, define, and instantiate an SWT" and "how two SWTs are associated by instantiating *rdfs:domain* relation". In order to generate seeds for *Google based meta-crawler*, a special full-text index is created to simulate Google's index and search mechanisms.

- The **Analysis** component hosts modular analytical tasks such as ranking SWDs and SWTs, summarizing the overall usage of SWTs, and classifying SWDs into ontologies and datasets.

- The **Search Services** component provides search and navigation services based on Swoogle's metadata. Nineteen REST web service APIs are specially developed to support machine agents' data access activities. A PHP based website is built on top of the Swoogle APIs to support human users as well as to test the APIs. The service APIs are highlighted by demonstrating the enhanced Search and Navigation model.

VI.B.2 Searchable Meta-Description for SWD and SWO

Swoogle consider the following categories of meta-description for a Semantic Web document: (i) *document metadata* which describes the general metadata of Semantic Web document as a Web document, e.g., document URL and last-modified time; (ii) *semantic content summary* which describes the metadata of the RDF graph of a Semantic Web document, e.g., the number of triples in the document; and (iii) *semantic content annotation* which extracts useful annotations about the Semantic Web document from the parsed RDF graph, such as *rdfs:comment* describing the document's URL. Note that Swoogle only store the compact metadata instead of the entire semantic content (i.e., RDF graph) of SWDs.

Document Metadata

For each candidate URL, Swoogle indexes the following document metadata:

- *url* - the URL
- *extension* - the filetype extension parsed from the URL, e.g., "rdf", "owl".
- *date-discover* - the date when the URL has been first discovered
- *date-ping* - the date when the URL has been last pinged (visited) by Swoogle's RDF crawler.
- *ping-state* - the result of last ping on the URL. The possible ping states are enumerated as the following:
  - *disallow* accessing the URL is disallowed by *robots.txt*
  - *cannot-connect* the URL cannot be connected
  - *response-forbidden* accessing the URL is forbidden according to HTTP response code
  - *response-redirected* - the URL is redirected to another URL according to HTTP response code
  - *cannot-download* - the content cannot be completely downloaded
  - *oversize* - the size of the document is too large so that our semantic parser cannot process it
  - *alive* the document is successfully downloaded and its content remain the same
  - *alive-modified* - the document is successfully downloaded but its content has been changed
- *last-modified* - the last-modified time as extracted from corresponding filed in HTTP header.
- *doc-md5sum* - the MD5 hash for the document

- *doc-length* - the number of bytes of the document
- *doc-charset* - the charset used to encode this document, e.g., “utf-8”
- *date-cache* - the date of latest cached version of the document

Semantic Content Summary

Swoogle indexes the following semantic content summary:

- *parse-state* - how the document has been parsed.
- *is-embedded* - whether this document is embedded
- *uses-grammar* - what kind of RDF syntax grammar, RDF/XML, NTriples or N3, is used by the SWD.
- *doc-triples* - the number of triples parsed from the SWD.
- *ontoRatio* - the percentage of definitional triples in this SWD.

Semantic Content Annotation

Full-text index on the RDF graph of an SWD is not suitable because it does not acknowledge the semantics of meta-usage of SWT. For example, consider two triples in *FOAF ontology*,

foaf:Person rdf:type rdfs:Class

foaf:Person rdfs:comments “A human being”

It is easy to see that the resource *foaf:Person* and the literal “A human being” are related to the topic of the ontology; however, the resources *rdf:type* and *rdfs:comments* are not. Hence, Swoogle collects *semantic content annotation* about the topic of a Semantic Web document from two sources: (i) the text annotation (about the document) which is extracted from triples having the document URL as their subject, and (ii) the local-name and text annotation of SWTs defined or referenced in the graph. In addition, Swoogle indexes *semantic content annotation* about the term-usage in a Semantic Web document from one source: the local-name of SWTs described in the graph.

VI.B.3 Searchable Meta-Description for SWT

Swoogle considers the following meta-descriptions for a Semantic Web term: (i) the term’s URI, namespace and local-name and (ii) the aggregated annotation.

URI, Namespace and Local-name

As described in [12], a URI consists of two parts: a namespace, which offers a unique Web address for the URI, and a *local-name*, which conveys the meaning. For example, the URI `http://xmlns.com/foaf/0.1/mbox` has the namespace `http://xmlns.com/foaf/0.1/` and the local-name *mbox*.

Since URIs are often quite long, many users encode the semantics of a Semantic Web term mainly in its local name, and search Semantic Web terms by their local-name. There are two practical issues in searching local-names.

First, special operations must be used to correctly extract local-name from URIs which do not use "#" to separate local-name and namespace. A widely used heuristic split a URI at the last back slash; hence, `http://xmlns.com/foaf/0.1/Person` can be split into two parts: namespace `http://xmlns.com/foaf/0.1/` and local-name *Person*. However, this heuristic may fail in some cases, e.g. the URI `http://foo.com/ex.owl` should not be decomposed. Therefore, Swoogle avoids this unsafe heuristic and adopts several other safe heuristics, e.g. using declared namespaces extracted during parsing the syntax of an SWD to identify the namespace component of a URI.

Second, although most local-names are simple, e.g., the top 40 frequently used local-names (see Table VI.1) are simple words about our daily life; there are still many in compound form, e.g., *number_of_wheels* or *GraduateResearchAssistant*. A simple regular expression pattern has been used to split such compound local-names into smaller pieces. The decomposition result and the original local-name together are used as the meta-description of the Semantic Web term.

Table VI.1: The 40 most used local-names (March 14, 2006)

name	#swt	name	#swt	name	#swt	name	#swt
name	883	head	242	language	201	lastName	162
Person	596	address	231	author	194	Class	155
title	457	source	220	Article	190	city	155
css	398	Animal	218	email	190	Resource	154
description	372	Event	216	Agent	173	Name	153
type	356	value	216	Country	173	state	148
location	338	country	214	Location	171	comment	147
date	309	year	212	number	166	Collection	145
Organization	260	Date	209	url	165	Conference	145
Book	247	Address	202	label	162	contains	145

Aggregated Annotation

Swoogle aggregates an SWT's literal descriptions, each of which is obtained from triples having the SWT as subject and a literal string as object. For example, it extracts "a human being" from a triple (foaf:Person rdfs:comment, "a human being"). The aggregated literal descriptions are used as the meta-description of the SWT.

Swoogle also aggregates the types of an SWT, which are extracted from meta-usages of the SWT from different SWDs. We may find incompatible meta-usages from different SWDs. For example, the term *rdfs:subClassOf* is well-known (according to RDFS ontology) an instance of *rdf:Property*; however, there exists one SWO[2] that has an incompatible meta-usage that defines the term as an instance of *rdfs:Class*. Swoogle enables users to search SWTs by *type* constraint: (i) the query "type:class" searches all SWTs being defined by*owl:Class*, *rdfs:Class* or *daml:Class*; (ii) "type:owl" searches all SWTs being defined by an SWT whose namespace is OWL; and (iii) "type:owl.class" searches all SWTs being defined by *owl:Class*.

VI.B.4 Swoogle's Metadata for Navigational Network

Swoogle maintains a global view of the navigational paths in the Semantic Web and thus forms a navigation network for the information consumers.

For each Semantic Web document, Swoogle records all namespaces used by the document (*path 1*), all related Semantic Web terms and their usage in the document (*path 3* and *path 8*), all referenced Semantic Web documents (*path 2*) and Semantic Web ontologies (*path 7*). Moreover, *paths 2 and 7* are bi-directional.

For each Semantic Web term, Swoogle records Semantic Web terms contributed to its definitions (*path 4*), and the namespace used by it (*path 5*). Moreover, *path 13 and 14* are reversely computed while *path 4* is bi-directional.

For each Semantic Web namespace, Swoogle records its official ontology derived from heuristics. *Paths 11 and 12* are reversely computed from the global catalog.

VI.C Swoogle Ranking

Ordering SWDs and SWTs is very important to sorting the search results as well as choosing the best one from several alternative candidates. A direct impact of ranking is that the emergence of the Semantic Web

[2] `http://www.w3.org/2000/10/swap/infoset/infoset-diagram.rdf`

will be reinforced by ranking: good Semantic Web documents and terms will be more visible and thus being better populated while bad ones will be disqualified when time elapses.

Intuitively, our ranking approximates the likelihood of a Semantic Web user surfing to an SWD, SWO or a SWT based on the current Semantic Web search and navigation model. Here, the Semantic Web surfing behaviors are not facilitated by Swoogle.

VI.C.1 Rational Surfing Model

In order to rank the popularity of Semantic Web documents, we introduce a *rational surfing model*: a rational surfer always recursively pursues the definitions of classes and properties to completely understand a given RDF graph. Figure VI.6 illustrates the rational surfing behavior.

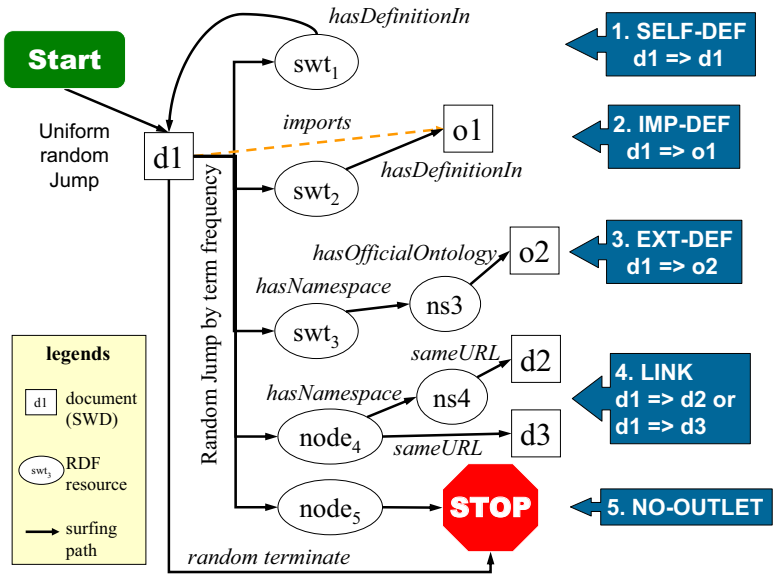


Figure VI.6: The rational surfing model

1. A rational surfer starts by jumping randomly to one of the accessible SWDs with uniform probability.

2. The surfer either terminates surfing with constant probability or chooses one RDF node in the RDF graph of the SWD, and the node is chosen based on its term frequency in the SWD's N-Triples version.

3. Once chosen an RDF node, the surfer either surfs to another document or terminates surfing based on the semantics of the chosen node:

(a) Path 1, 2 and 3 acknowledge the pursuing definition behavior: if the node is not anonymous and has class or property usage in the present document, the surfer pursues its definition from the present document (path 1 SELF-DEF), the imported ontologies (path 2 IMP-DEF), or the ontology addressed by the namespace parted of the node's URI (path 3 EXT-DEF).

(b) Path 4 (LINK) shows the hyper-link based surfing behavior: if the node is not anonymous and does not have class or property usage, the surfer follows the URL obtained from its URI or namespace to another Semantic Web document.

(c) Path 5 (NO-OUTLET) includes all cases when no further surfing path starts from the present node, e.g. the present node is literal or anonymous, or the present node's URI links to a normal Web document.

VI.C.2 Rational Ranks of SWDs and SWTs

Based on the *rational surfing model* explained above, we first compute *rational ranks* of Semantic Web documents using a variation of PageRank [123], and then compute the rational ranks of Semantic Web terms. Let D be the set of SWDs to be ranked, the *rational ranks* are derived by the following steps:

1. **initialize DocRank.** $DocRank$ is a vector that stores the *rational ranks* of all SWDs. For each SWD d, its rational rank $DocRank(d)$ indicates the likelihood of a rational surfer surfing to d. Since the rational rank is iteratively computed, and $DocRank^{(i)}$ refers to the $DocRank$ derived by the i-th iteration. For each SWD d, we initialize its $DocRank$ using Equation VI.1.

$$DocRank^{(0)}(d) = 1 \qquad \forall d \in D \tag{VI.1}$$

2. **derive DocRankNav.** The likelihood of accessing an SWD d via the existing navigation network in the Semantic Web is derived from Equation VI.2, which is the sum of in-flows from other SWDs scaled by a constant dumping factor α (an empirical value 0.85 is used in our experiments).

$$DocRankNav^{(i+1)}(d) = \alpha \times \sum_{s \in S(d)} DocRank^{(i)}(s) \times NavFlow(s, d) \tag{VI.2}$$

where $S(d)$ is the set of Semantic Web documents that have navigational paths to SWD d, and $NavFlow(s, d)$ is the probability of the rational surfer surfing from SWD s to SWD d. $NavFlow(s, d)$ is derived as

the normalized number of non-stop navigational paths (i.e., whose types are among paths 1 to 4 as illustrated in Figure VI.6) as shown in Equation VI.3.

$$NavFlow(s,d) = \frac{PathCount(s,d)}{3 \times n(s)} \tag{VI.3}$$

where $PathCount(s,d)$ is the number of non-stop navigational paths from SWD s to SWD d via a resource in s, and $n(s)$ is the number of triples in s. Note that we count navigational paths based on the N-Triples version of SWD s: each resource or literal in the N-Triples version contributes one navigational path. So there can be as many as $3 \times n(s)$ navigational paths starting from SWD s, and this number is chosen as the normalize factor. In practice, $NavFlow(s,d)$ are pre-computed and reused in each iteration.

3. **derive DocRankRand.** Besides accessing an SWD via navigation network, a rational surfer can directly jump to the SWD randomly. We assume that (i) all SWDs should be accessed in the same likelihood and (ii) the surfer must jump to a new SWD once it stops surfing. Therefore, we dynamically adjust $DocRankRand(d)$, i.e., the likelihood of the rational surfer randomly jumping to an SWD d, using Equation VI.4, which uniformly distributes the sum of flows via *path 5 (STOP)* to each SWD. Note that $||x||_1$ is the *1-norm* of a vector x.

$$DocRankRand^{(i+1)}(d) = 1 - \frac{||DocRankNav^{(i+1)}||_1}{|D|} \tag{VI.4}$$

4. **derive new DocRank.** The new DocRank of an SWD d is then derived by Equation VI.5 that sums $DocRankNav$ and $DocRankRand$. The assignment of $DocRankNav$ in step 3 ensures that the *1-norm* of $DocRank^{(i+1)}$ is the same as the *1-norm* of $DocRank^{(i)}$.

$$DocRank^{(i+1)}(d) = DocRankRand^{(i+1)}(d) + DocRankNav^{(i+1)}(d) \tag{VI.5}$$

5. **test convergence.** We iteratively compute $DocRank$ until converge, i.e., the difference between $DocRank^{(i+1)}$ and $DocRank^{(i)}$ is lower than a constant number ε as shown in Equation VI.6. If the difference is still large, we will go back to *step 2*; otherwise, we stop iteration and call the current $DocRank$ vector $DocRank^{(converged)}$. Note that the *termination threshold* ε is empirically set to 0.01.

$$\sum_{k=1}^{N} \left| DocRank^{(i-1)}(d_k) - DocRank^{(i)}(d_k) \right| < \varepsilon \tag{VI.6}$$

6. **derive TermRank** Once the rational ranks of SWDs have converged, we further compute rational ranks of SWTs using the Equation VI.7. The rank of an SWT t depends on two components: (i) $DocRank^{converged}(s)$, i.e., whether the SWD s that describes t is highly ranked; and (ii) $\frac{TC(s,t)}{3 \times n(s)}$, i.e., whether the SWT s has been frequently used by SWD s. Moreover, when there are significant amount of SWDs using the SWT t, t can also be highly ranked.

$$TermRank(t) = \sum_{s \in D} \frac{TC(s,t) \times DocRank^{converged}(s)}{3 \times n(s)} \tag{VI.7}$$

where $TC(s,t)$ is the term frequency of SWT t in SWD s.

VI.C.3 Evaluation using Controlled Examples

When no SWD has in-link, the navigation network does not exist and there is no need to propagate $DocRank$. Therefore, all SWDs' rational ranks are "1" because they can only be accessed via random jump.

When the navigation network is not empty, we study the following representative examples in Figure VI.7. The rank processes of these examples run at most thirteen iterations when the termination threshold ε is set to 0.001; therefore, the convergence of algorithm is verified. In these examples, the ranking results are easy to be explained.

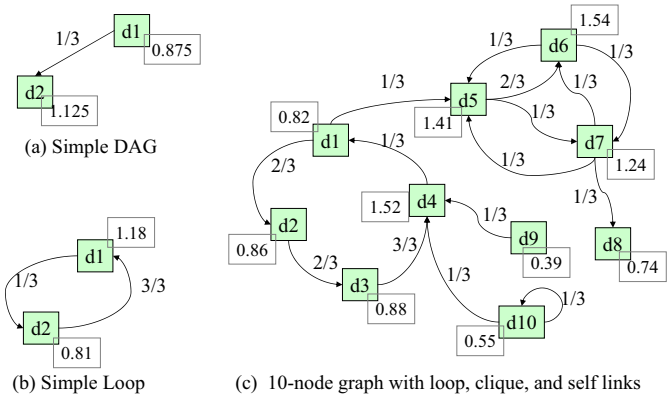


Figure VI.7: Example ranking results using rational surfering model.

The shaded block denotes SWD, the dotted-edge block denotes the final $DocRank$ of the corresponding SWD, the tag "1/3" on an arc denotes that one out of a total of three navigational paths of the source SWD links to the pointed SWD.

- *DAG*, see Figure VI.7-a. When SWDs are interlinked as a DAG (directed acyclic graph), the one with in-flow is ranked higher than the one without in-flow.

- *two-node cycle*, see Figure VI.7-b. When two nodes are interconnected and form a loop, the one with higher in-flow will be ranked higher.

- *multiple-node loop and clique*, see Figure VI.7-c. According to Equation VI.2, the rank of node in loop or clique depends on both the in-flow (i.e., $NavFlow(s, d)$) and the rank of nodes linking to it (i.e., $DocRankNav(s)$). For example among $d5, d6$ and $d7$, node $d6$ is ranked the highest because (i) nodes (i.e., $d1$, $d6$, $d7$) linking to $d5$ do have not consistent high $DocRank$ and (ii) $d7$ only has in-flow from two nodes (i.e., $d5$ and $d6$).

- *self-link*, see Figure VI.7-c. Self-link makes node $d10$ ranked higher than $d9$ even if they have the same out-flow. This observation acknowledges the rational surfers' preference in finding ontologies: more ontology usages (as indicated by in-flow) are preferred.

VI.C.4 Evaluation using Real World Data – *SW06MAR*

We also evaluate ranking algorithm using *SW06MAR* which has about 1.2 million of SWD. Figure VI.8 depicts the top 10 highest ranked SWDs and the significant flow between them. We have the following observations:

- It is notable `http://purl.org/dc/terms` (dcTerms ontology) is ranked high simply because it is linked by another popular ontology `http://pur.org/dc/elements/1.1/` (dc ontology) even though its own in-degree and mean-inflow[3] are both low.

- `http://www.w3.org/2000/01/rdf-schema` (RDFS ontology) is ranked higher than `http://www.w3.org/1999/02/22-rdf-syntax-ns` (RDF ontology) because of the unequal flow between them (the flow from the latter to the former is 0.35; and the flow from the former to the latter is only 0.11).

Consequently, we derive the ranking of SWTs. Table VI.2 lists the twenty best ranked SWTs with the number of SWDs and the number of their occurrences in navigation path. The SWTs are ordered by "rank"(i.e., TermRank of the SWT). For each SWT, the column "#swd" shows the number of SWDs that

[3]The *in-degree* and the *mean-inflow* is derived using all non-stop navigational paths linking to the destination SWD. The former counts all such paths, and the latter is the mean of the normalized flow of these paths.

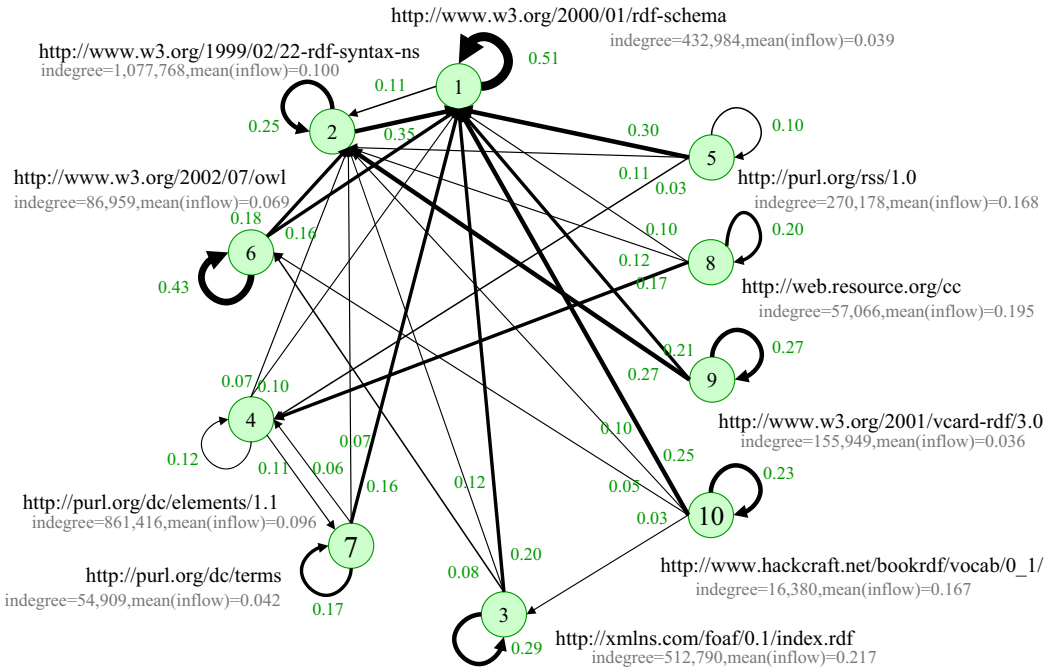


Figure VI.8: Ranking results of top 10 highest ranked SWDs in *SW06MAR*

An oval denotes an SWD and the number in the oval denotes the SWD's order of rank. Directed arc indicates the flow from one SWD to another SWD, and the thickness (and the optional labels) of an arc is determined by the strength of the flow.

have at least one meta-usages of the SWT, and the column "#occurence" sum up the occurrence of all meta-usages of the SWT based on all SWDs. We have the following interesting observations:

- `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` is the most commonly used SWT.
- *rdfs:isDefinedBy* is ranked high because it has been used by many important SWOs even though it is rarely used by other SWDs. Moreover, it is ranked higher than *rdfs:domain* because it is more frequently use in non-stop navigational paths than the latter.
- *dc:title* and *foaf:Person* have been highly ranked due to their popularity.
- The SWTs having the same namesapce are usually ordered by popularity, which usually depends on the number of SWDs contributing the corresponding navigation path; therefore, *foaf:nick* is ranked higher than *foaf:knows* even though it has less occurrences.

Table VI.2: Top 20 highest ranked SWTs

URI of SWT	#swd	#occurence	rank
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	1,076,554	35,460,710	94,676
http://www.w3.org/2000/01/rdf-schema#label	32,711	2,243,214	26,710
http://www.w3.org/2000/01/rdf-schema#comment	15,672	745,915	24,304
http://www.w3.org/2000/01/rdf-schema#isDefinedBy	2,652	62,673	23,963
http://purl.org/dc/elements/1.1/title	601,717	12,225,947	23,388
http://www.w3.org/1999/02/22-rdf-syntax-ns#Property	57,749	511,823	22,410
http://www.w3.org/2000/01/rdf-schema#domain	6,783	174,533	13,042
http://www.w3.org/2000/01/rdf-schema#Class	7,800	254,958	12,950
http://www.w3.org/2000/01/rdf-schema#range	6,470	160,300	12,942
http://xmlns.com/foaf/0.1/Person	468,218	11,046,085	12,378
http://www.w3.org/2000/01/rdf-schema#seeAlso	398,439	10,756,684	11,523
http://www.w3.org/2000/01/rdf-schema#Resource	1,905	14,864	10,899
http://madskills.com/public/xml/rss/module/trackback/ping	170,351	365,206	10,360
http://purl.org/dc/elements/1.1/identifier	171,412	245,303	10,223
http://www.w3.org/2000/01/rdf-schema#subClassOf	8,900	1,799,970	9,498
http://xmlns.com/foaf/0.1/nick	360,053	9,895,830	8,012
http://xmlns.com/foaf/0.1/knows	350,532	10,441,736	7,787
http://xmlns.com/foaf/0.1/weblog	345,500	8,623,706	7,335
http://purl.org/rss/1.0/link	268,987	4,271,025	7,208
http://xmlns.com/foaf/0.1/interest	164,322	10,529,848	7,185

VI.D Summary

In this chapter, we have built two conceptual models for Semantic Web surfing: namely the *current Semantic Web search and navigation model* and the *enhanced Semantic Web search and navigation model*.

The *enhanced model* is supported by the Semantic Web search engine *Swoogle*. The most important contribution of Swoogle is its rich metadata about the Semantic Web, which significantly facilitates the navigation and search experience.

The *current model* leads to the rational surfing based algorithm that ranks the popularity of Semantic Web documents and terms. The ranking algorithm is not merely a variation of PageRank; indeed, it is based on Swoogle's metadata (especially the navigation network) and is highlighted by its explainable ranking results.

Chapter VII

PROVENANCE OF SEMANTIC WEB KNOWLEDGE

The Semantic Web is not simply a web of facts but indeed a web of belief because not all knowledge on the Semantic Web is trustworthy. Upon receiving a piece of knowledge (usually encoded in an RDF graph), the consumers may need to judge its trustworthiness before adding it to knowledge base. This chapter presents a two-step hypothesis-testing solution: (i) to find all supporting evidences throughout a collection of knowledge sources (usually the entire Semantic Web); and (ii) to aggregate the trustworthiness of individual sources into the overall trustworthiness of the hypothesis using provenance information.

VII.A Finding Supporting Evidence at Molecular Level

VII.A.1 The Semantic Web Knowledge Onion

The Semantic Web can be thought of as one large "universal" RDF graph distributed across many Web pages. Since the graph is an unwieldy view, we usually work with online Semantic Web documents. Semantic Web document is a natural and appropriate level of granularity for most tasks but still too coarse for finding supporting evidence. The finest granularity of Semantic Web knowledge is triple, which corresponds to each of the directed edges in RDF graph. There are also some other methods for grouping triples, e.g., *Named graph* [26] and *Concise Bounded Description* [147]. Figure VII.1 depicts the levels of granularity of RDF graph from the universal one to triples.

The *molecule* level is added because other levels of granularity are not appropriate to capture the "molecular" semantics: some triples cannot be further split without loss of information. For example, if there are

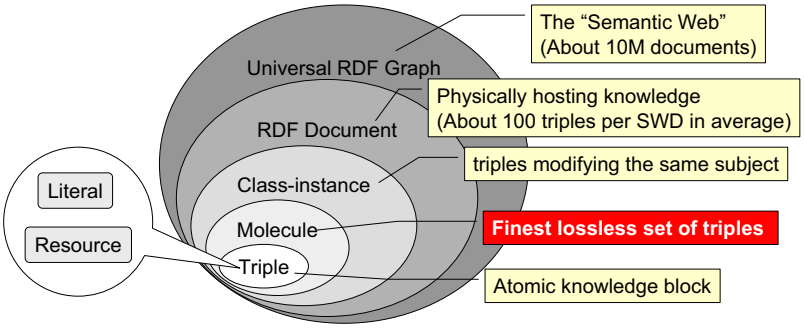


Figure VII.1: Levels of granularity of Semantic Web knowledge

two triples connected by a blank node asserting that “There is someone who has first name ‘li’ and last name ‘ding’ ”, splitting them into two separate triples (i.e., “there is someone whose first name is ‘li’ ” and “ there is someone whose last name is ‘ding’ ”) will loss the binding semantics conveyed by the blank node.

Apparently, the molecular level of granularity is driven by the usage of *blank node* in RDF graph. The semantics of blank nodes in RDF graphs has been studied in different application contexts, including F-logic inference [160], signing entire graphs [25] and minimal self-contained graphs [153], tracking changes of graphs [11] and definitions of resources [147], and tracking knowledge provenance [41]. All these works recognize the dual semantics of blank nodes: (i) existential semantics, i.e., showing the existence of an individual instead of explicitly referencing the individual, and (ii) binding semantics, i.e., showing the inherent relations between triples and usually capturing the representation of complex relations via simple binary-relation provided by RDF graph.

Based on these insights, the molecular level granularity is needed because triple level granularity only captures the existential semantics but misses the binding semantics of blank node.

VII.A.2 Supporting Evidence

A piece of knowledge in the Semantic Web is normally encoded in an RDF graph; therefore, the *supporting* relation between two RDF graphs is essentially the why-provenance.

Definition 7 (supporting evidence) . *Given two RDF graphs a and b, b is called the supporting evidence of a if at least one* molecular *sub-graph of a can be inferred from b. The supporting evidence can be* complete *if b implies a or* partial *if b is not complete but does imply a sub-graph of a. The molecular level granularity is required to preserve the binding semantics which is critical to the meaning of an RDF graph.*

Figure VII.2 illustrates several scenarios in finding the supporting evidence of a hypothesis RDF graph. Note that a blank-node free triple is always a molecular sub-graph, otherwise more discussion is needed.

- Since $G1$ implies $G3$, we may assert that $G1$ is a complete supporting evidence for $G3$.
- Both $G2$ and $G3$ are partial supporting evidence to $G1$ because $t3, t4, t5$ can be inferred from $G2$ and $t1$ can be inferred from $G3$.
- $G4$ cannot be a supporting evidence to $G1$ unless $t3$ can be asserted as a molecular sub-graph of $G4$.

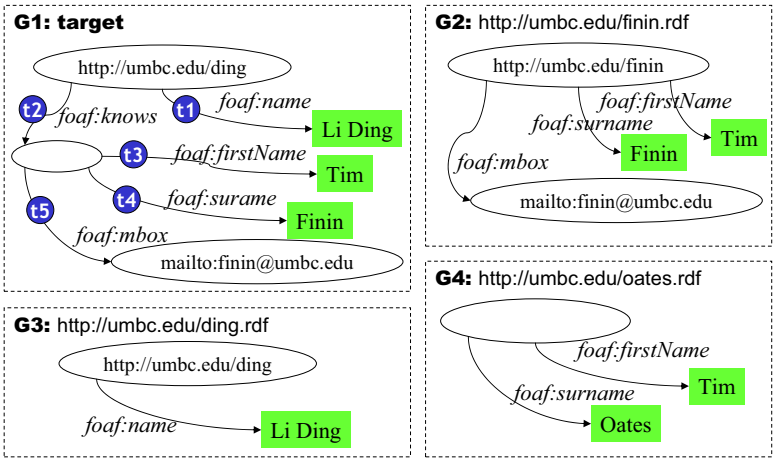


Figure VII.2: Example RDF graphs that motivate molecular level granularity

$G1$ has five triples asserting that a thing with URI "http://umbc.edu/ding" and name "Li Ding" knows a thing with name "Tim Finin" and mbox "mailto:finin@umbc.edu".

Suppose each of the four RDF graphs in Figure VII.2 is stored in a Semantic Web document. Neither $G2$ nor $G3$ completely supports $G1$; hence, the document level of granularity may end up with low "recall" due to the miss of partial supporting evidence. So do *Named Graph* and *Concise Bounded Description* based approaches. The triple level of granularity, on the other hand, could mistakenly assert $G4$ being partial supporting evidence to $G1$ because a triple only preserves the *existential semantics* but ignores the consequence of triples being bounded by virtue of sharing the same blank node.

VII.A.3 RDF Graph Decomposition

In order to find supporting evidences, we must first decompose the hypothesis RDF graph into molecular sub-graphs.

Definition 8 (RDF graph decomposition) *An RDF graph decomposition has three elements (W, d, m): the background ontology W, the* **decompose** *operation $d(G, W)$ which breaks an RDF graph G into a set of sub-graphs $\hat{G} = \{G_1, G_2, ..., G_n\}$ using W, and the* **merge** *operation $m(\hat{G}, W)$ which combines all $\hat{G}$'s elements into a unified RDF graph G' using W[1]. In addition, $\hat{G}$ is called a* **partition** *of G if all elements in $\hat{G}$ are disjoint.*

In RDF graph decomposition, it is important to address node equivalence problem, i.e., whether two nodes in an RDF graph are equivalent or not. The "official" guideline is provided in RDF [79] and OWL [35] specifications: non-anonymous nodes in an RDF graph can be combined if they share the same URI; but merging anonymous nodes depends on the semantics provided by the background ontology. Berners-Lee and Connolly [11] mentioned the *functionally grounded* blank nodes, which can be compared using the semantics of *owl:InverseFunctionalProperty* (IFP) and *owl:FunctionalProperty* (FP). By running a little bit forward chain inference in the background ontology, such IFP and FP semantics can be further propagated via *owl:inverseOf* and *rdfs:subPropertyOf*. In addition, equivalence semantics such as *owl:sameAs*, *owl:equivalentClass* and *owl:equivalentProperty* can be used to compare related blank nodes as well.

Usually the nodes in an RDF graph G can be classified into three disjoint groups: U for RDF nodes with URIrefs, L for literals, and B for BNodes (i.e., blank RDF node). Based on the above observations, we can classify RDF nodes into another three disjoint groups:

- A node n is **naturally grounded** (or grounded) if n is in either U or L.
- A node n is **functionally grounded** according to the background ontology W if n is in B and any of the following conditions is met:
    1. there exists a triple (n, p, o) in G, p is *IFP* according to W, and o is either grounded or functionally grounded.
    2. there exists a triple (s, p, n) in G, p is *FP* according to W, and s is grounded or functionally grounded.
    3. there exists a node n' in G such that n is equivalent to n' according to W, and n' is grounded or functionally grounded.

[1]Note that we restrict our discussion in the context where no inferred triples are produced, and we adopt *RDF graph equivalence* semantics from RDF [95].

- Given an RDF graph G with background ontology W, a node n in G is said **contextual grounded** if n is in B but n is not *functionally grounded*.

It is notable that a node n could be functionally grounded on multiple RDF nodes, e.g., when both *foaf:homepage* and *foaf:mbox* are confirmed as *IFP* according to background FOAF ontology, an instance of *foaf:Person* could functionally grounded on the homepage, the email, or both.

VII.B Lossless RDF Graph Decomposition and RDF Molecule

An RDF graph decomposition (W, d, m) is **lossless** when we can reconstruct the RDF graph without adding or losing any information using the decomposition result, i.e., when for any RDF graph G, $G = m(d(G, W), W)$. The lossless property is very important since this allows us to manipulate RDF graph at a finer level of granularity – RDF molecule.

Definition 9 (RDF molecules) *RDF molecules of an RDF graph G are the finest, lossless sub-graphs obtained by a lossless decomposition (W, d, m) on G. Here, a sub-graph is* lossless *if it can be used to restore the original graph without introducing new triples, and it is the* finest *if it cannot be further decomposed into lossless sub-graphs.*

Based on the definition of molecule and the classification of RDF nodes in the previous section, we identify three basic types of RDF molecules:

Terminal Molecule (T-molecule). A *terminal molecule* only contains grounded nodes and functionally grounded BNodes. All BNodes in T-molecule are "closed". A BNode bn in a molecule m is called "closed" if it is functionally grounded and used by exactly two triples in m, otherwise it is "open".

Non-Terminal Molecule (NT-molecule). A *non-terminal molecule* only contains grounded nodes and at least one functionally grounded BNodes. Only one of the BNodes is "open", and it is called the *active-functionally-grounded node*[2]. Intuitively, an *NT-molecule* is the path in RDF graph that makes a BNode functionally grounded.

Contextual Molecule (C-molecule). A *contextual molecule* contains grounded nodes and at least one *contextual grounded* BNode(s). A C-molecule is *maximum* if it is not sub-graph of any other *C-molecules* in G. Maximum contextual molecules are the only possible *C-molecules* in lossless decomposition.

[2]Note that functionally grounded BNodes in "open" state cannot co-exist in the same molecule.

VII.B.1 Naive Lossless RDF Graph Decomposition

We start with the naive decomposition without using background ontology. The corresponding *decompose* operation is essentially an algorithm that computes connected components. Note that only arcs connecting two blank nodes are of our interest according to the definition of molecules. It produces a *partition* of the present RDF graph with well-known time complexity – approximately O(V+E) for an RDF graph with V nodes and E arcs. A straightforward algorithm includes the following steps:

1. break a graph into a set of sub-graphs, each of which contains only one triple,

2. merge sub-graphs which share the same blank node until no more merge can be done.

Such decomposition produces only *T-molecules* and *C-molecules*. The decomposition can be demonstrated in Figure VII.3: the first result molecule (t1) is a T-molecule since both its subject and object are in U or L; the second result molecule that consists of (t2,t3,t4,t5) is a C-molecule since they share the same BNode $?x$. This decomposition is lossless since triples connected by BNodes are kept together.

```
{t1} (http://www.cs.umbc.edu/~dingli1  foaf:name "Li Ding")
{t2} (http://www.cs.umbc.edu/~dingli1  foaf:knows   ?x )
{t3} (?x  foaf:name  "Tim Finin")
{t4} (?x  foaf:mbox  "finin@umbc.edu")
{t5} (?x  foaf:mbox  "finin@cs.umbc.edu")
```

Figure VII.3: Example of Naive lossless RDF graph Decomposition

The five-triple graph asserts that a foaf person with foaf name “Tim Finin” and two mboxes “finin@umbc.edu” and “finin@cs.umbc.edu” is known by the foaf person with mbox “dingli1@umbc.edu” and a foaf name“Li Ding”.

VII.B.2 Functional Lossless RDF Graph Decomposition

A **Functional decomposition** refines the result of a naive decomposition using functional dependencies asserted by the background ontologies. In this case, the semantics of *owl:InverseFunctionalProperty* (IFP), *owl:FunctionalProperty* (FP), and OWL’s same-as properties can be used to label blank nodes with corresponding peers’ URIs, and the semantics of *owl:inverseOf* and *rdfs:subPropertyOf* can be used to propagate the labeling results.

With background ontology, we can figure out functionally grounded nodes and thus reduce the size of *C-molecules* by splitting triples grouped by functionally grounded BNodes. The corresponding decompose operation $d_f(G, W)$ is shown as the following:

1. Create a molecule for each triple in G and label the type of the molecule;

2. Generate NT-molecules using functional dependencies in G according to W;

3. Generate new T-molecules by combining two different NT-molecules sharing the same *active-functionally-grounded node*.

4. Generate new molecules by combining an existing C-molecule cm with an NT-molecule ntm when ntm's active-functional-grounded node *afgn* is used by cm but not functionally grounded in cm, and then remove cm if ntm is a new C-molecule. Repeat this step until no new molecules are generated.

5. For each BNode bn in G which is not used by any of the NT-molecules of G, generate a new molecule ncm by combining all C-molecules links to or from it, and then remove those C-molecules (since they all are sub-graph of ncm). At the end of iteration, all the residual C-molecules are *maximum C-molecules*.

The above operation $d_f(G, W)$ generates all possible molecules for G given background ontology W.

Consider the example shown in Figure VII.4. This graph asserts that the (unique) person who has mbox "dingli1@umbc.edu" also has a first name "Li" and a surname "Ding". The addition of the assertion about the *foaf:mbox* functionally grounds the blank node designated by $?x$ since this property is defined as an "inverse functional" property in the background ontology. The graph can be decomposed into two molecules, one with the mbox and name triples and another with the mbox and surname triples. The blank nodes in each molecule can be renamed, yet we are still able to combine the two molecules and reconstruct the original graph. A notable observation is that the molecules with name assertion are often found in much more web knowledge sources than the other molecules.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
{t1} (?x  foaf:name   "Li Ding")
{t2} (?x  foaf:surname "Ding")
{t3} (?x  foaf:mbox   "dingli1@umbc.edu")
```

Figure VII.4: Example for Functional lossless RDF graph decomposition

The three-triple graph asserts that the unique foaf person with foaf mbox "dingli1@umbc.edu" also has a foaf name "Li Ding" and a foaf surname "Ding". There are two molecules: (t1,t3) and (t2,t3).

By applying a background ontology, which specifies that *foaf:mbox* is an IFP, over the RDF graph in Figure VII.3, the result includes six T-molecules: (t1), (t2,t4), (t3,t4), (t2,t5), (t3,t5), and (t4,t5), plus two NT-molecules: (t4), (t5). Note that the molecule (t2,t3) is not recognized as a T-molecule or an NT-molecule because it has *contextual grounded* BNode $?x$, and it is not recognized as a C-molecule because $?x$ could

be functionally grounded due to $\{$t4$\}$. The number of generated molecules can be much greater than the number of triples because molecules are generated as a combinational result and they could be redundant to one another. However, molecules are smaller in size and do enumerate all finest possible information blocks conveyed by the original RDF graph. This feature is extremely important to exhaust all possible (partial) evidence for the original RDF graph.

Finally, Figure VII.5 shows a more complicate situation where a graph contains two blank nodes. It is not decomposable using naive approach, but with functional decomposition, the blank node identified by $?y$ is functionally grounded by the combination of triples $t3$ and $t5$. Hence, this graph can be decomposed into three molecules: (t1,t3), (t2,t3), and (t3,t4,t5).

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix kin: <http://ebiquity.umbc.edu/ontologies/kin/0.3/>.
{t1} (?x  foaf:firstName   "Li")
{t2} (?x  foaf:surname "Ding")
{t3} (?x  foaf:mbox   "dingli1@umbc.edu")
{t4} (?y  foaf:surname "Wang")
{t5} (?y  kin:motherOf ?x)
```

Figure VII.5: Another Example for Functional lossless RDF graph decomposition

The five-triple graph asserts that a foaf person with surname Wang is the mother of the unique foaf person with foaf mbox "dingli1@umbc.edu", foaf firstName "Li", and foaf surname "Ding".

VII.B.3 Heuristic Lossless RDF Graph Decomposition

Heuristic decomposition studies blank nodes which can be uniquely identified by a set of properties according to domain-specific heuristics in some applications, e.g., *foaf:firstName* and *foaf:surname* together can be used to identify a person. Intuitively, this is essentially the "key" concept widely used in database literature. We could call this *heuristic grounding* to distinguish it from *functional grounding*.

Such heuristics are common for many applications including natural language processing (e.g., in co-reference resolution), information extraction from text (e.g., named entity recognition) and mailing list management (e.g., identifying duplicates). There is a rich literature of approaches to this problem, ranging from work on databases [51] to recent work involving the Semantic Web [65]. Consider the example in Figure VII.6. Our heuristic might be that knowing either (i) a person's name and home phone number or (ii) a person's name and home address, is sufficient to uniquely identify a person[3]. Using this heuristic, this graph can be decomposed into three molecules: (t1,t2,t3), (t1,t2,t4) and (t1,t3,t4).

[3]This is a heuristic that will fail sometimes, as is the case of Heavyweight Boxer George Foreman and his sons

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
{t1) (?x  foaf:name   "Li Ding")
{t2} (?x  foaf:homePhone "410-555-1212")
{t3} (?x  foaf:homeAddress "1000 Hilltp Circle, Baltimore MD 21250")
{t4} (?x  foaf:age "27")
```

Figure VII.6: Example for Heuristic RDF graph decomposition

Using a heuristic rule, we identify three molecules in this graph: (t1,t2,t3), (t1,t2,t4) and (t1,t3,t4).

VII.B.4 Molecular Decomposition Results on Harvested SWDs

We have sampled several Semantic Web documents to evaluate RDF molecule decomposition. By randomly selecting SWDs from our sample dataset, we have collected five representative example groups. Table VII.1 lists the example SWDs and corresponding decomposition results by counting the triples and molecules derived.

Table VII.1: Molecular decomposition results on selected examples

URL of SWD	# triple	# T-m	# NT-m	# C-m
10 highest ranked SWDs				
http://www.w3.org/2000/01/rdf-schema	87	87	0	0
http://www.w3.org/1999/02/22-rdf-syntax-ns	85	85	0	0
http://xmlns.com/foaf/0.1/index.rdf	563	563	0	0
http://purl.org/dc/elements/1.1	146	146	0	0
http://purl.org/rss/1.0/schema.rdf	44	44	0	0
http://www.w3.org/2002/07/owl	160	155	0	5
http://purl.org/dc/terms	604	604	0	0
*http://web.resource.org/cc	66	66	0	0
http://www.w3.org/2001/vcard-rdf/3.0	138	138	0	0
http://www.hackcraft.net/bookrdf/vocab/0_1/	54	54	0	0
5 ontologies with blank nodes				
http://swrc.ontoware.org/ontology	576	304	0	68
http://www.w3.org/2002/03owlt/testOntology	80	47	0	7
http://www.w3.org/2002/12/cal/ical	1,482	532	0	160
http://www.daml.org/2001/10/html/airport-ont	39	11	0	7
http://inferenceweb.stanford.edu/2004/07/iw.owl	480	240	0	56
5 randomly selected FOAF documents				
http://blogs.dion.ne.jp/neko_no_ehon003/foaf.rdf	66	0(89)	0(12)	1(0)
http://prophetseye.typepad.com/foaf.rdf	7	0 (15)	0(3)	1(0)
http://www.wasab.dk/morten/2005/01/photos/5/image-06.rdf	294	191(382)	0(13)	17(51)
http://minetaka.com/hgl/foaf.rdf	10	0(17)	0(2)	1(0)
http://ajft.org/2004/05/16/206-0628_img.rdf	31	25(27)	0(1)	2(3)
5 randomly selected RSS documents				
http://washi.over-blog.com/index.rdf	90	78	0	1
http://blog.livedoor.com/xml/common_theme_cat6.rdf	2,107	1,805	0	1
http://donut.lv3.net/test/rss.cgi/lunge/1090177455.rdf	409	357	0	1
http://blog.lib.umn.edu/guent006/agp/index.rdf	10	8	0	1
http://yaplog.jp/mayayan/index1_0.rdf	54	47	0	1
5 randomly selected other SWDs				
http://www.w3.org/2001/Talks/05www10-swfig/rdfg3b.rdf	80	80	0	0
http://bitsko.slc.ut.us/chatlogs/echo/2003-09-03.rdf	504	218	0	72
http://www.bndfc.co.uk/whats-new/arts-culture/.meta.rdf	35	19	0	4
http://amk.ca/books/h/Samurais_Wife.rdf	7	7	0	0
http://iw4.stanford.edu/proofs/laptop/ex9.owl	28	21	0	1

* this is an embedded SWD.

TOP 10 SWDs Top 10 SWDs are indeed well populated ontologies. Most only use terms from RDF schema and do not have blank nodes (except OWL ontology). Therefore, the number of T-molecules is the

same as the number of triples because each triple is indeed a T-molecule.

Ontologies with Blank Nodes Although many ontologies are free of blank nodes, the ontology constructs provided by OWL do encourage the use of blank nodes and thus result in considerable number of C-molecules. For example, the Inference Web ontology[4] contains 480 triples and can be decomposed into 240 T-molecules, each of which has only one triple, and 56 C-molecules, each of which has four (e.g., for *owl:Restriction* on cardinality) to eleven triples (e.g., the object list of *owl:unionOf*). We have found non-trivial number of this kind of SWOs according to their SWT usage: *owl:Restriction* (331,730 occurrences), *owl:intersectionOf* (52,090 occurrences), *owl:unionOf* (12,286 occurences), and *owl:oneOf* (6,145 occurrences).

FOAF documents FOAF documents are unique because both naive decomposition and functional decomposition could be applied according to FOAF ontology. In particular, the C-molecules can be further decompose when they use Inverse Functional Properties such as *foaf:mbox*, *foaf:homepage*. The number of molecules after functional decomposition is usually smaller than the number of triples, but exceptions exist.

RSS documents RSS documents have a regular decomposition pattern – many T-molecules and only one C-molecule. The C-molecule is usually an instance of *rss:items* that links to a *rdf:sequence* of *rss:item* instances.

Other SWDs FAOF and RSS documents have contributed significant number of SWDs; however, there are still some SWDs belonging to neither of them. The background ontologies of these SWDs seldom provide functional dependency definition, hence only T-molecules and C-molecules are found in those SWDs.

VII.C Provenance based Trust Aggregation

Once we have obtained the supporting evidences from multiple sources on the Semantic Web, a trust model is need to (i) interpret the impact and trustworthiness of each individual source and then (ii) aggregate their trustworthiness into the final trustworthiness of the hypothesis. For example (see Figure VII.7) we may collect evidences from six sources: “CIA World Fact Book”, “NASDAQ”, “FOO News”, “Agent K”, “agent

[4]This ontology can be found at http://inferenceweb.stanford.edu/2004/07/iw.owl.

W", and "Department of State". In general, "Department of State", "CIA World Fact Book" and "NASDAQ" are highly trusted, the two agents are trusted differently, and "FOO News" is somewhat less trusted.

Having merged the knowledge obtained from the six sources, an analyst then finds a semantic path from "Mr. X" to "Osama Bin Laden" in Figure VII.7. Based on her trust in the sources and contribution made by each sources, the analyst need to derive an overall trust in the semantic path. The situation may be more complex when "Agent K" and "FOO NEWS" have conflicting beliefs over the molecule "Organization B invests Company A".

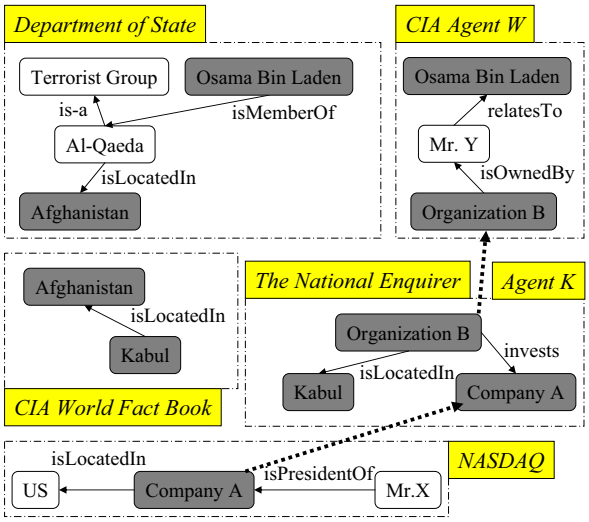


Figure VII.7: An example that needs trustworthiness evaluation

VII.C.1 Modeling and Bootstrapping Belief and Trust

Our study adopts the truthful and consistent semantics for trust, i.e., a set of molecules are trusted by an agent only when they are consistent and believed to be true. We focus on two important associations in trust study: *belief*, which shows an agent's trust in a particular set of molecules, and *trust* which shows an agent's trust in another agent's knowledge in whole. The value of trust and belief characterizes the trustor's confidence over the trusted object. Our literature survey [42] has found a full spectrum of valuation scheme for quantifying trustworthiness, and this study chooses a real number between [0,1] to represent trust value where 1 stands for complete trustworthy, 0.5 stands for neutral or default opinion, and 0 stands for completely untrustworthy.

The **belief** assertion can be derived from *whom-provenance*. For example, *NASDAQ* fully believes *(Company A, isLocatedIn, US)* because it is the publisher of *http://example.com/nasdaq.rdf* which contains the

molecule.

The **trust** assertion can often be derived in two steps. First, some common-sense rules are employed to extract social relations from raw Semantic Web data. For example, { *(X is_author_of P), (Y is_author_of P), (X is not Y)* } implies coauthor relation between X and Y. A more complicated social relation is "neighborOf(X,Y)" which we might deduce from { *(X livesIn C),(Y livesIn C), (C rdf:type Street),(X is not Y)*. Second, users may apply personalized rules to infer trust (including confidence value) from the extracted or existing social networks. Similarly, we may derive strong trust from "Li Ding" to "Tim Finin" using coauthor information obtained from DBLP since most publications of the former person have the latter person as coauthor, and our heuristic rules assert that high ratio of co-authorship implies strong collaboration relations and thus strong trust relation. Moreover, we may derive the default trust between unfamiliar parties using well known reputation system, e.g., the default trust in "Tim Finin" could be derived from Google's PageRank value on his homepage.

VII.C.2 Trusting Hypotheses Supported by Multiple Sources

Given a set of molecules extracted from the Semantic Web, how much should we trust in the model they describe? This can be viewed as a problem central to document analysis in which not all information sources are trusted at the same degree.

We model the solution to this issue using the following notation: $S = \{s_1, s_2, ..s_n\}$ be a collection of n RDFS molecules, Z be the analyst agent, $T(x, y)$ be the trust value from agent x to agent y, $B(x, s)$ be how agent x believes in molecule s, $TH(x)$ be the set of highly trusted agents by x, $C(x, S)$ be the trustworthiness of a collection of molecules S according to agent x, $source(s)$ be the collection of agents that are the provenance of a molecule s, $AB(s)$ be the collection of agents who have belief states on a molecule s.

Simple Belief Aggregation

First we study a simple situation in which molecules are independent, semantically consistent, and fully believed by their source (i.e., $B(x, s) = 1, \forall x \in source(s)$). We build a Bayesian probabilistic model of knowledge sources based on "Noise-Or" theory [128]. Equation VII.1 assumes that provenance agents are independent and that their knowledge accuracy is correctly captured as trust. Therefore, we use "Noise-OR" method to derive the probability that molecule s_i is true to agent A given A's trust to the provenance agents

$Source(s_i)$. This followed by a multiplication which aggregates the overall confidence since all molecules are independent.

$$C(Z,S) = \prod_{s_i \in S} \left(1 - \prod_{x \in source(s_i)} (1 - T(Z,x))\right) \quad \text{(VII.1)}$$

Using this technique, if analyst Z's trust in NASDAQ is $T(Z, NASDAQ) = 0.99$, Z's trust in "FOO NEWS" is $T(Z, FOO) = 0.5$, Z's trust in "Agent K"is $T(Z, K) = 0.6$, Z's trust in "CIA Agent W" is $T(Z, W) = 0.8$, then $C(Z, S0)$, where S0 refers to the semantic path from "Mr.X" to "Osama Bin Laden" (as mentioned in introduction section), is $0.99 \times (1 - (1 - 0.5)(1 - 0.6)) \times 0.8 \cong 0.63$. This path is much more trustworthy than the cases that only one of "Agent K" and "FOO NEWS" is the provenance agent.

Aggregating Conflicting Beliefs from Trusted Peers

The second situation is more complicated since (i) inconsistent molecules may be detected according to ontological semantics (e.g., a person's name can not have two different values) and (ii) more agents beside the provenance agents may assert beliefs through RDF graph reference. A straightforward approach is consensus model which is based on the intuition that trusted peers' opinions are the only sources of reliable information. Equation VII.2 averages the discounted belief confidence from trusted agents.

$$C(Z,S) = \prod_{s_i \in S} \left(\sum_{x \in AB(s_i) \cap TH(Z)} \frac{T(Z,x) * B(x,s_i)}{|AB(s_i) \cap TH(Z)|}\right) \quad \text{(VII.2)}$$

Assume that the molecule s_1 " Organization B invests Company A" is believed by "FOO NEWS" (i.e., $B(FOO, s_1) = 0.9$) but disbelieved by "Agent K" (i.e., $B(K, s_1) = 0.1$). According to the analyst Z, $C(Z, \{s1\})$ is $(0.9 \times 0.1 + 0.5 \times 0.9)/2 = 0.54$ when $T(Z, K) = 0.9$ and $T(Z, FOO) = 0.5$, and is $(0.5 \times 0.1 + 0.5 \times 0.9)/2 = 0.5$ when $T(Z, K) = 0.5$ and $T(Z, FOO) = 0.5$. In both case, s_1 is not trustworthy to Z and more field investigation is needed; however, the first case should be investigated in higher priority due to its higher trust value.

VII.D Summary

This chapter addresses data quality issues in Semantic Web data access. In particular, it presents methods for finding supporting evidence of a hypothesis RDF graph in the Semantic Web and aggregating the overall trustworthiness of the hypothesis using provenance information.

Chapter VIII

SEMANTIC WEB APPLICATION MODELS

What is the application model of the Semantic Web, and what is the business model of the Semantic Web? These questions are commonly posted and discussed in the Semantic Web community [13, 104, 52]. Based on the work presented in previous chapters, we have built some interesting applications of the Semantic Web. In this chapter, we describe these applications and explain their importance and interestingness.

VIII.A Semantic Web Search Engine

Although the Semantic Web is on the Web, no conventional search engines substantially help users to search and browse the Semantic Web. Therefore, a Semantic Web search engine by itself is an important application of the Semantic Web.

Semantic Web search engine plays an important role in Web-scale Semantic Web data access. It works better than any of conventional Web search engines in searching the Semantic Web because it only indexes Semantic Web data and provides effective search interface for searching Semantic Web data. It better supports Semantic Web surfing than existing Semantic Web browsers and editors (e.g., HyperDAML[1] and Swoop[2]) because of its denser navigation network, richer navigation path semantics, and global view of the Semantic Web.

Semantic Web search engine is also useful for studying the Semantic Web surfing behaviors modeled by the *enhanced Semantic Web search and navigation model*. We build the *Swoogle 2006* website as the demonstration of Semantic Web search engine, and use it to study users' surfing behaviors. In the rest of this

[1] http://www.daml.org/2001/04/hyperdaml/
[2] http://www.mindswap.org/2004/SWOOP/

section, we illustrate the human and machine interfaces of our Swoogle Semantic Web search engine, and show how Swoogle is different from conventional Web search engine in finding ontologies.

VIII.A.1 Semantic Web Surfing using Swoogle APIs and Website

In order to enhance users' search and navigation experiences in the Semantic Web, we have developed two user interfaces: (i) the REST web service interface that consists of 19 Swoogle APIs for machine agents and (ii) the human friendly Website (`http://Swoogle.umbc.edu`) for human users. The services are based on the sample dataset *SW06MAR* and the metadata generated from it. The website is built on top of the 19 Swoogle APIs and can be viewed as a machine agent that translates the results (encoded in RDF/XML) returned by APIs into HTML.

Since the end of January 2006 when *Swoogle 2006* was officially released, Swoogle website has received a total of 77,697 valid hits[3] by 7,078 human users (including 127 registered users[4]) from over 107 countries or regions. Moreover, Swoogle APIs has logged 118,875 API calls[5]. These overall statistics reflects considerable interest in Swoogle from the Semantic Web community.

In Table VIII.1, we briefly explain the meaning of Swoogle APIs. For each API, we count the number of API calls (i.e., "#hits") and the number of unique queries (i.e., "#unique") to judge its usefulness and interestingness. In particular, *search* services have been accessed much more than other services for three main reasons:

- *Search* services are located on the front page of the website while the rest services require at least one-step navigation inside the website.
- Users are not yet familiar with Swoogle APIs or the enhanced search and navigation model so that they use the search services a lot without noticing the existence of digest and navigation services.
- Users may repeat the same query to retrieve the complete result set because the number of unique queries of an API is much less than the number of API calls of it. For example, many users assumed that they could find instances of a certain class or namespace using document search service (search_document); however, the best way to achieve this goal is using term-to-document navigation service (rel_swt_swd).

[3] We have filtered out about 40,000 meta-crawling hits by a un-friendly robot

[4] We promote user registration by allowing registered users to view more than 100 query results.

[5] Unfortunately, we cannot filter the meta-crawling hits in this statistics. We did notice that meta-crawling mainly called the search_swt_all service significant times for each query string to exhaust corresponding search results

Table VIII.1: Swoogle API usage (March 23, 2006)

ID	API name	#hits	#unique	meaning of API
search services				
1	search_swd_all	87,676	1,765	list documents matching document search
2	search_swd_ontology	22,406	6,900	list ontologies matching ontology search
3	search_swt	3,917	1,354	list terms (URI references) matching term search
digest services				
4	digest_swd	1,904	651	show document's metadata
5	digest_swt	1,731	790	show term's metadata
6	digest_ns	281	78	show namespace's metadata
navigation services				
7	rel_swd_swt	377	108	list terms mentioned in this document
8	rel_swt_swd	333	103	list documents related to this term
9	rel_swd_swd_to	249	93	list documents linking to this document(in-link)
10	rel_swd_ns	228	91	list namespaces mentioned in this document
11	rel_swd_swd_from	220	97	list documents linked by this document(out-link)
12	rel_swt_swt_from	186	99	list terms defining this term (out-link)
13	rel_swt_swt_to	180	94	list terms defined by this term (in-link)
14	rel_swd_instance_range_p2c	136	79	terms (properties) which are used as the range of the present term in instance data
15	rel_ns_swd	106	28	list documents using this namespace
16	rel_swd_instance_domain_p2c	88	63	terms (properties) which are used as the domain of the present term in instance data
17	rel_swd_instance_range_c2p	85	68	terms (properties) which use the present term as range in instance data
18	rel_ns_swt	83	22	list terms using this namespace
19	rel_swd_instance_domain_c2p	80	67	terms (properties) which use the present term as domain in instance data

A follow-up experiment studies popular Swoogle query strings to understand the distribution of users' interests in using Swoogle services. Since most of the top 50 Swoogle queries are using either the *search_swd_all API* or the *search_swd_ontology API*, we split the observed 12,176 distinct Swoogle queries into three groups, and list the popular queries in each group to detect interesting usage patterns (see Table VIII.2, and the column "hits" records the number of API calls). Our investigation shows that popular queries are often promoted by repeated access to Swoogle API to obtain the complete search results. The top two *search_swd_all* queries are drawn by a meta-crawler that aims at exhausting the result set.

Many popular queries in Table VIII.2 are related to FOAF ontology. We guess they are the results of a frequently encountered sequence of Semantic Web surfing activities:

1. searching "person" ontology using search_swd_ontology API. This action acknowledges the first scenario (*fining ontologies*) discussed in Chapter I (see Figure VIII.1).

2. viewing metadata of the selected *FOAF ontology* using digest_swd API.

3. listing SWDs that link to the selected *FOAF ontology* using rel_swd_swd_to API (see Figure VIII.2).

Table VIII.2: Popular Swoogle queries (March 23, 2006)

Query Type	Query string	#hits
Top 5 queries using search_swd_all		
search_swd_all	local	18,605
search_swd_all	foaf	14,560
search_swd_all	organization	1,708
search_swd_all	people	1,521
search_swd_all	friend	1,491
Top 5 queries using search_swd_ontology		
search_swd_ontology	rdf	558
search_swd_ontology	car	385
search_swd_ontology	service	366
search_swd_ontology	person	348
search_swd_ontology	food	325
Top 10 queries using other query types		
digest_swd	http://xmlns.com/foaf/0.1/index.rdf	186
rel_swt_swd	http://xmlns.com/foaf/0.1/Person	144
search_swt	service	140
digest_swt	http://xmlns.com/foaf/0.1/Person	121
search_swt	person	102
digest_swd	http://www.cyc.com/2004/06/04/cyc	101
rel_swd_swd_to	http://xmlns.com/foaf/0.1/index.rdf	83
rel_swd_swt	http://xmlns.com/foaf/0.1/index.rdf	83
search_swt	food	83
digest_swt	http://purl.org/dc/elements/1.1/creator	75

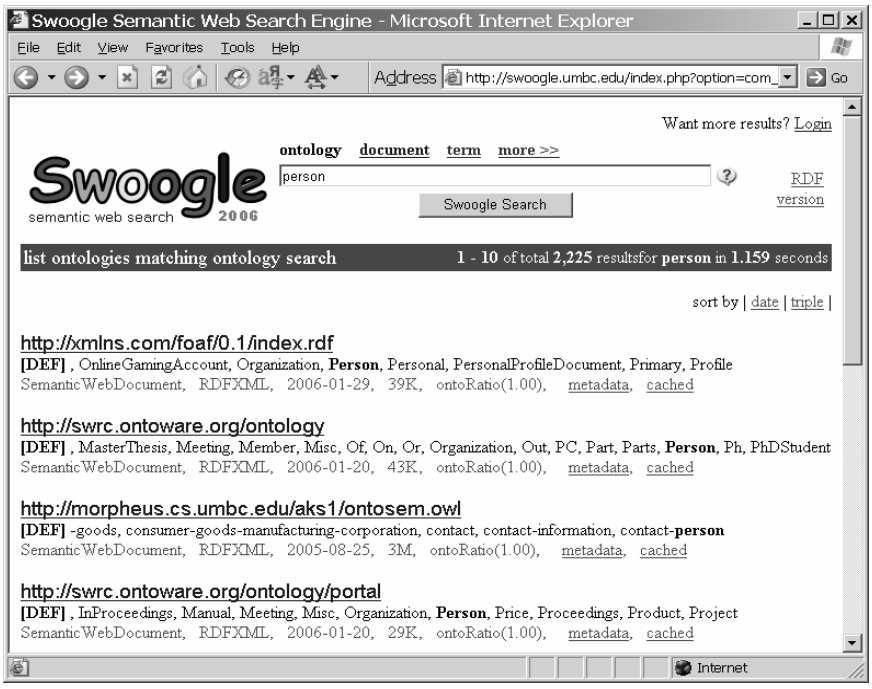
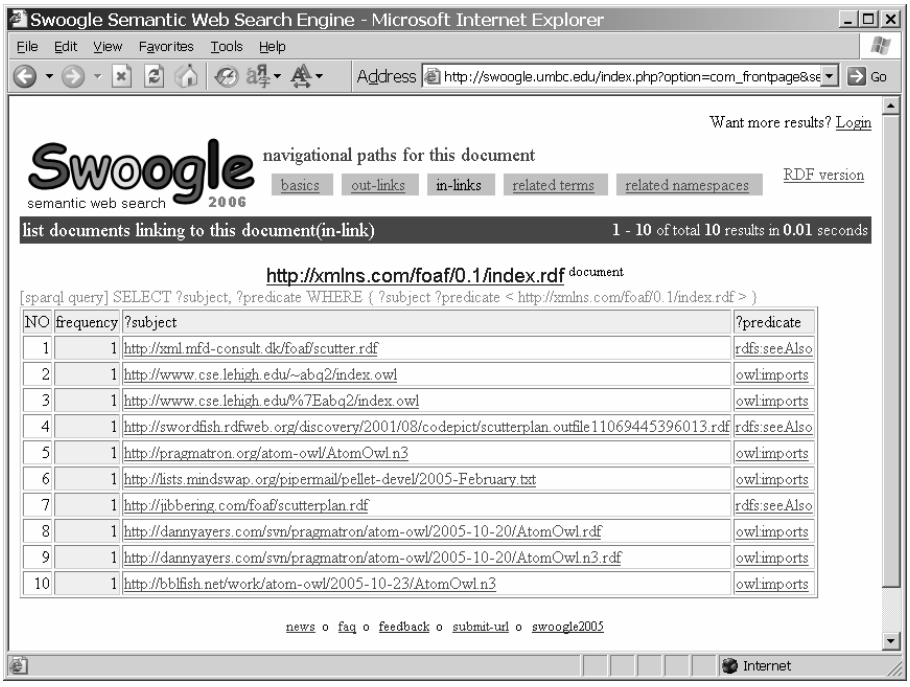


Figure VIII.1: Searching for “person” ontologies using Swoogle

This action acknowledges the second scenario (*enumerating inverse links of owl:imports*) discussed in Chapter I.



(a) the HTML version

(b) the RDF/XML version

Figure VIII.2: Finding SWDs Linking to the present SWD

4. listing the meta-usages of SWTs in the selected *FOAF ontology* using rel_swd_swt API (see Figure VIII.3).

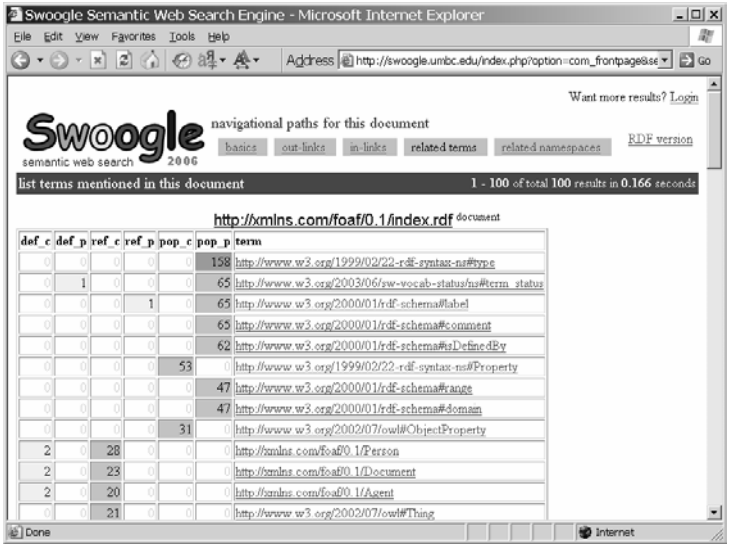


Figure VIII.3: Finding SWT usages in the present SWD

5. viewing metadata of the SWT *foaf:Person* using digest_swt API.

6. listing all SWDs containing instances of foaf:Person using rel_swt_swd API (see Figure VIII.4). This action acknowledges the third scenario (*enumerating all instance documents*) discussed in Chapter I.

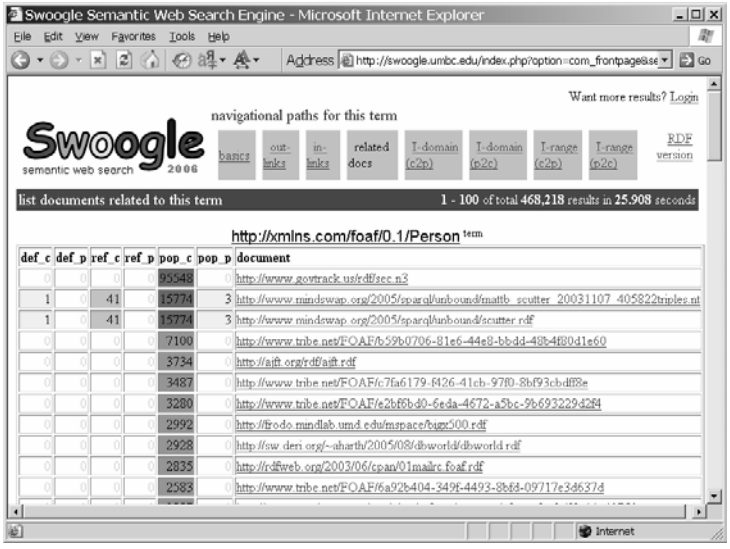


Figure VIII.4: Finding SWDs populating instances of *foaf:Person*

VIII.A.2 Finding Ontologies Using Semantic Web Search Engine

We cannot always rely on Web search engines to find ontologies. For example, when searching for “person” ontology, we may try Google queries such as “person ontology”, “person filetype:owl”, “person filetype:rdf”, and “person rdf” and expect FOAF ontologies being returned in the first 10 results; however, these queries never meet our expectations. Moreover, most documents returned by conventional search engines are not Semantic Web documents.

Unlike the conventional Semantic Web search engines, Swoogle indexes only Semantic Web documents so that all search results are Semantic Web documents. Moreover, it collects a comprehensive meta-description about the semantic content and structure of a Semantic Web document based on the parsed RDF graph. By default, Swoogle searches all relevant ontologies as long as an SWO matches at least one of the following features: URL, text description, and local-names of the defined or referenced Semantic Web terms. For example, the query string “person” will return all relevant ontologies mentioning the term “person” even when it only appears in their URLs. Moreover, Swoogle supports feature-specific queries, for example, users may search ontologies that have more than 10,000 triples using query string “hasCntTriple:[10000 to 1000000]”, or search Semantic Web documents that has instantiated class related to “person” using query string “pop:person”.

Based on the access log, we study users’ interests in finding ontologies by analyzing the query strings used in *search_swd_ontology* API. Swoogle queries received high number of hits are often submitted by a few users and is not effective in evaluating the popularity of the queries among all users; therefore, our study focuses on the diversity of query string to Swoogle APIs. Our study has found 6,686 unique query strings submitted to *search_swd_ontology* API, and this number indicates a wide range of interests in using Swoogle to find ontologies.

Table VIII.3 shows the randomly selected 50 query strings submitted to *search_swd_ontology* API. Most query strings only have several hits. Interestingly, many of them are ontology related terms, names, popular nouns and verbs. These queries may be used to drive the future development of Semantic Web ontologies.

Table VIII.3: 50 random query string submitted to *search_swd_ontology* API

query	#hits	query	#hits
url:"http://www.w3.org/2000/01/rdf-schema"	6	woman man	1
pablo	2	forall	6
"rank" "search engine"	1	politics	4
documentation	1	asp.net	1
division bell	1	er	2
nutch	1	ontology editor	4
clothes	3	naive bayes	1
house	28	koala	14
munich	1	book + amazon	1
Individual	3	900k	1
LRI-Core	1	agriculture l3s	1
vocabulary registry	3	s95	1
food vegetable	1	mass	1
MI6	2	teach	1
recette fiscale	1	danny ayers	1
nevatie	2	bible	4
surgery	2	locations	1
ELECTRI EQUIPMENT	1	como se llama jaime?	1
langue	1	knowledge economy	1
cashflow	2	elephant	4
script	1	amsterdam	2
lotus	1	find	2
thailand	2	web semantica	1
university ontology	4	'http://xmlns.com/foaf/0.1/index.rdf'	2
how to get the search engine	1	concept	4

VIII.B Semantic Web Archive Service

Semantic Web archive service is an add-on service that exposes the cached versions of SWDs to the information consumers. The archive service follows the idea of the Internet Archive[6]. We have observed three interesting usages of the archive service:

- We can track the evolution of a Semantic Web ontology. For example, the Protege ontology has been updated at least five times during 2005 (see Figure VIII.5), and these versions are good real world examples for ontology evolution study [120, 119].

- We can track the growth of instance data, for example, the changes of a FOAF document. The number of versions and the time span between each version help us to determine whether a Semantic Web document is actively maintained and contains fresh data. For example, the SWD shown in VIII.6 has been actively maintained during 2005 and its size is growing. We may further investigate if the publisher has modified instance space, the schema space, or both, and find interesting usages of Semantic Web

[6] `http://www.archive.org/`

```
http://protege.stanford.edu/plugins/owl/protege

About this URL
The latest ping on [2006-01-29] shows its status is [Succeed, changed into SWD].
Its latest cached original snapshot is [2006-01-29 (3373 bytes)]
Its latest cached NTriples snapshot is [2006-01-29 (41 triples)].

We have found 7 cached versions.

2006-01-29: Original Snapshot (3373 bytes), NTriples Snapshot (41 triples)
2005-08-25: Original Snapshot (3373 bytes), NTriples Snapshot (41 triples)
2005-07-16: Original Snapshot (2439 bytes), NTriples Snapshot (35 triples)
2005-05-20: Original Snapshot (2173 bytes), NTriples Snapshot (30 triples)
2005-04-10: Original Snapshot (1909 bytes), NTriples Snapshot (28 triples)
2005-02-25: Original Snapshot (1869 bytes), NTriples Snapshot (27 triples)
2005-01-24: Original Snapshot, NTriples Snapshot (31 triples)
```

Figure VIII.5: Tracking evolution of the Protege ontology

ontologies and patterns of Semantic Web data, for example, the evolution of social network [6].

```
http://www.csee.umbc.edu/~dingli1/foaf.rdf

About this URL
The latest ping on [2006-01-29] shows its status is [Succeed, changed into SWD].
Its latest cached original snapshot is [2006-01-29 (6072 bytes)]
Its latest cached NTriples snapshot is [2006-01-29 (98 triples)].

We have found 6 cached versions.
2006-01-29: Original Snapshot (6072 bytes), NTriples Snapshot (98 triples)
2005-07-16: Original Snapshot (6072 bytes), NTriples Snapshot (98 triples)
2005-06-19: Original Snapshot (5053 bytes), NTriples Snapshot (80 triples)
2005-04-17: Original Snapshot (3142 bytes), NTriples Snapshot (50 triples)
2005-04-01: Original Snapshot (1761 bytes), NTriples Snapshot (29 triples)
2005-01-24: Original Snapshot, NTriples Snapshot (29 triples)
```

Figure VIII.6: Tracking evolution of an FOAF document

- We can learn the life cycle of Semantic Web documents, especially the static ones. Usually a static Semantic Web document is actively maintained for a while after it has been published on the Web, then its update frequency may decrease or increase during a long period, and finally it will go offline or move to another Web location. For example, the SWD shown in Figure VIII.5 was created, updated and removed on the Web during 2005. This study is useful since most users assume the Semantic Web document should not be frequently changed or be moved from its original Web location.

```
http://simile.mit.edu/repository/fresnel/style.rdfs.n3

About this URL
The latest ping on [2006-02-02] shows its status is [Failed, http code is not 200 (or406)].
Its latest cached original snapshot is [2005-03-09 (15809 bytes)]
Its latest cached NTriples snapshot is [2005-03-09 (149 triples)].

We have found 3 cached versions.
2005-03-09: Original Snapshot (15809 bytes), NTriples Snapshot (149 triples)
2005-02-25: Original Snapshot (12043 bytes), NTriples Snapshot (149 triples)
2005-01-26: Original Snapshot, NTriples Snapshot (145 triples)
```

Figure VIII.7: Tracking the life cycle of a Semantic Web document

This archive service mainly uses Swoogle's cached data (4,949,019 versions of SWDs occupying approximately 300G disk space) and a corresponding cache log table in database that associate each pair of (URL of SWD, cached date) with a cached file.

VIII.C Swoogle Ontology Dictionary

Swoogle Ontology Dictionary is another add-on application on top of Swoogle. It collects all Semantic Web terms from the harvested Semantic Web documents and builds a global view of the Semantic Web vocabulary. It has two potential contributions to the Semantic Web community:

- It builds a comprehensive view of the Semantic Web vocabulary and breaks the (unnecessary) physical boundary imposed by Semantic Web ontologies. There are two well-known drawbacks of using ontology documents to group Semantic Web terms: (i) Semantic Web terms defined in one Semantic Web ontology may be instantiated in quite different frequencies, for example, *owl:versionInfo* is far less instantiated than *owl:Class* in the Semantic Web; and (ii) Semantic Web terms from multiple ontologies are usually used together to modify one class-instance, for example, *rdfs:seeAlso* and *dc:title* have been frequently used together to modify the class-instances of *foaf:Person*.

- Beside the Semantic Web terms defined or referenced in Semantic Web ontologies, it also collect the Semantic Web terms which have been instantiated as classes or properties but have not been defined by any existing Semantic Web ontology. For example, the property `http://webns.net/mvcb/generatorAgent` has been widely used, and interested users may want to reuse this term even though no existing Semantic Web ontology has defined it.

VIII.C.1 Searching and Browsing Semantic Web Terms

Currently, Swoogle ontology dictionary provides two user-interfaces for locating Semantic Web terms.

- *Term Search* is essentially a web-interface based on Swoogle term search API, which allows users to search SWTs by URI, namespace, local-name, literal definitional description, and semantic definition.

- *Alphabetical Term Index*, as shown in Figure VIII.8, organizes all Semantic Web terms by prefix alphabetically. It has two views: the *prefix view* (left panel) and the *matched-term-list view* (right panel).

In the prefix view, each prefix is followed by the number of terms using that prefix[7]. In the matched-term-list view, all terms matching current prefix are listed.

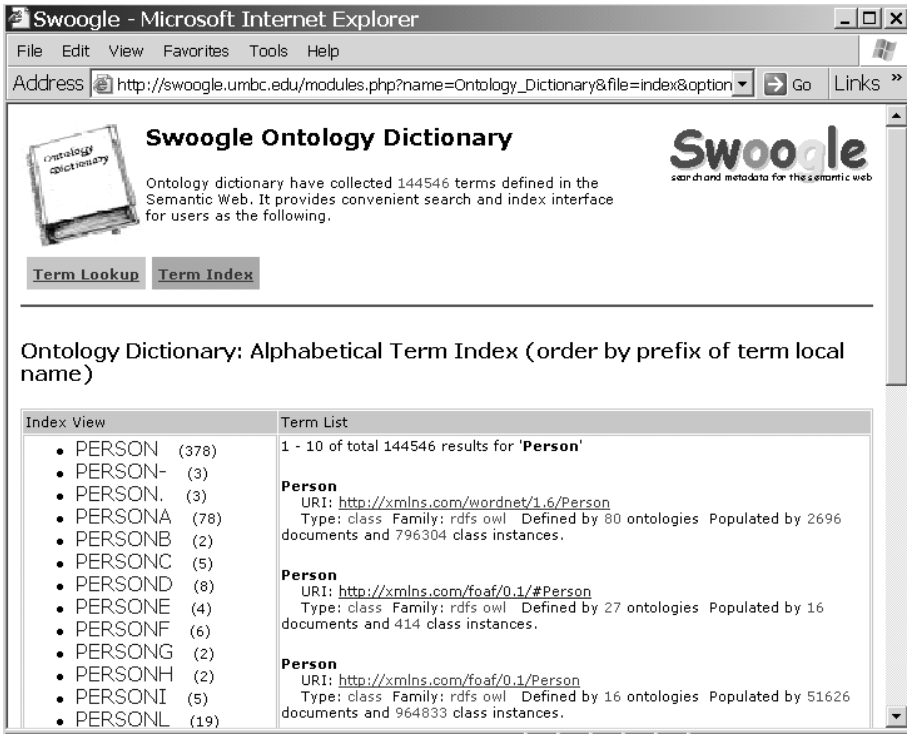


Figure VIII.8: The alphabetical term index interface

VIII.C.2 Aggregated Definition of Semantic Web Terms

Once a Semantic Web term has been found, Swoogle ontology dictionary shows the user an aggregated definition of the term based on the global view of the Semantic Web, including definitions obtained from Semantic Web ontologies and the definitions of *rdfs:domain* and *rdfs:range* instantiated in instance data.

Figure VIII.9 shows that the definition of *foaf:Person* could be obtained from three sources including two Semantic Web ontologies and one Semantic Web document. Swoogle ontology dictionary considers the following definitions of a Semantic Web term:

- **term metadata** (see ONTO 2 in Figure VIII.9). The metadata includes the basic Swoogle's metadata about the term, i.e., namespace, local-name, and the aggregated definition triples (triples rooted at the present SWT) obtained from all ontologies harvested. We notice that a term could be declared as a

[7]We use case-insensitive string matching here.

class and a property at the same time by different ontologies. For example, `http://www.w3.org/2000/01/rdf-schema#range` has been defined as a property by RDFS ontology and a class by a Semantic Web ontology `http://infomesh.net/2001/rdfsvalid/inv-example.n3`.

- **domain and range definition** (see ONTO 1 in Figure VIII.9). Note that we additionally include domain and range definition in the unified definition of a Semantic Web term because users may need to know the *instance-properties* (i.e., properties that can be used to modify the instance of the class) of a class. Knowing this kind of definitions helps users to instantiate class-instances as well as run RDFS inference.

- **instantiated domain and range definition** (see SWD 3 in Figure VIII.9). Besides the definition provided by Semantic Web ontologies, we may learn the domain and range definition from the triples using *rdfs:domain* or *rdfs:range* as predicate in instance data. Such instantiated domain and range definition can be used as a guideline for evaluating the utility of corresponding definitions in Semantic Web ontologies. For example, when *dc:title* has been frequently used as the *instance property* of *foaf:Person*, we may add the corresponding *rdfs:domain* definition to FOAF ontology as the result of reverse-engineering.

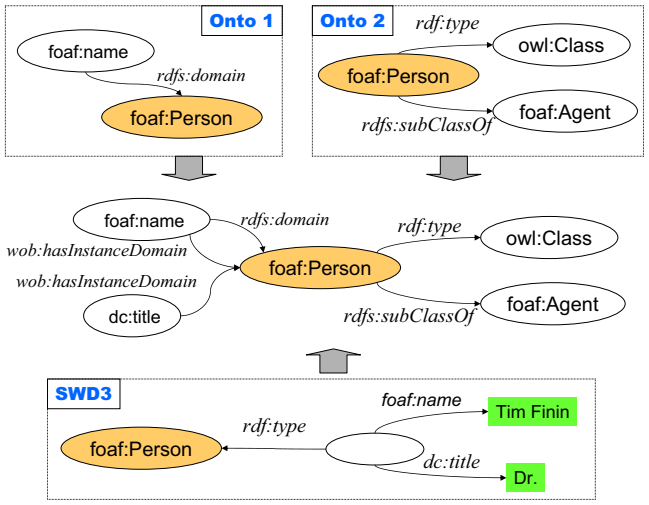


Figure VIII.9: Aggregated term definition

VIII.C.3 Ontology DIY and Reengineering

Swoogle Ontology Dictionary provides a flexible interface for users to access Semantic Web terms: (i) users can directly search for Semantic Web terms instead of searching for relevant Semantic Web ontologies first; (ii) it offers user unified definitions of Semantic Web terms obtained from the meta-usage of the terms distributed on the Semantic Web.

Although Semantic Web ontologies are widely used to organize a set of related Semantic Web terms, Swoogle Ontology Dictionary offers users another way to manage ontologies – "Do It Yourself" (DIY). The dictionary enables users to compose their own ontologies using popular Semantic Web terms from multiple ontologies instead of merging those ontologies. A direct benefit of this approach is that irrelevant Semantic Web terms in imported ontologies are not included in the customized ontology. A good use-case is to build DIY ontology for *frame-based* representation: a user may create a DIY ontology by choosing some desired classes (i.e., frames) and then adding popular *instance-properties* (i.e., slots) of these classes. The DIY ontology indicates the popular practices among the publishers and promotes the emergence of common ontologies. The DIY ontology also helps information consumers to figure out the common instance structures.

Another important usage of Swoogle Ontology Dictionary is reengineering ontologies. Intuitively, the practices of instantiating Semantic Web terms in instance data greatly help ontology engineers to review and update the existing ontologies. For example, we may evaluate a term's popularity by counting the number of *wob:hasClassInstanceIn* navigational paths, and evaluate the domain or range relation between two terms by counting the number of the corresponding *Swoogle:hasInstanceDomain* and *Swoogle:hasInstanceRange* navigational paths. Based on the usage information, we may find only a few terms in a big ontology (e.g. CYC ontology) has been actually well instantiated, and find some term definitions (e.g. domain or range definition) that are used without being defined.

We have recently encountered an interesting case that aims at revising the Dublin Core Element ontology by enumerating the instantiated domain and range usages of properties in Dublin Core element ontology. Some of the results are listed in Table VIII.4. It is easy to see that *rdfs:Literal* is the most used range of all Dublin Core properties. Moreover, we have observed the trend of switching from literal description to structure data: the range of *dc:creator* has been instantiated as *foaf:Agent*, *wordnet16:Agent* and *foaf:person* in practice.

Table VIII.4: Most instantiated range relations of Dublin Core terms

property	range	#swd	#instance
dc:title	rdfs:Literal	433,091	843,259
dc:description	rdfs:Literal	400,052	2,484,990
dc:date	rdfs:Literal	264,253	5,231,875
dc:creator	rdfs:Literal	234,655	2,477,665
dc:language	rdfs:Literal	207,382	341,020
dc:subject	rdfs:Literal	173,686	2,941,474
dc:rights	rdfs:Literal	45,368	153,451
dc:publisher	rdfs:Literal	43,751	173,864
dc:publisher	http://www.hackcraft.net/bookrdf/vocab/0_1/Publisher	16,369	16,369
dc:format	rdfs:Literal	13,183	287,664
dc:identifier	rdfs:Literal	12,936	148,095
dc:language	http://purl.org/dc/terms/RFC1766	12,775	15,395
dc:format	http://purl.org/dc/terms/IMT	12,518	14,498
dc:identifier	http://purl.org/dc/dcmitype/Text	11,547	15,123
dc:rights	http://web.resource.org/cc/Agent	8,652	17,275
dc:contributor	rdfs:Literal	7,613	70,097
dc:rights	foaf:Person	5,282	12,691
dc:creator	http://web.resource.org/cc/Agent	4,090	6,359
dc:type	rdfs:Literal	3,903	62,182
dc:source	rdfs:Literal	3,390	86,949
dc:creator	http://xmlns.com/wordnet/1.6/Person	2,714	1,138,250
dc:creator	foaf:Person	2,281	5,969
dc:source	http://web.resource.org/cc/Work	1,925	1,925
dc:creator	foaf:Agent	1,723	3,234
dc:coverage	rdfs:Literal	1,186	6,438

VIII.D Semantic Web Data Access Applications

Semantic Web documents, in comparison with conventional Web documents, are better structured for machine consumption. In most cases, the collection of online Semantic Web document can be better viewed as a knowledge base or database distributed on the Web. In order to build useful applications by accessing Semantic Web data distributed on the Web, two requirements should be met: (i) there are enough Semantic Web data on the Web; and (ii) there exist effective tools that facilitate accessing the desired Semantic Web data on the Web. Our works address both requirements by harvesting online Semantic Web data, measuring the global properties, supporting Semantic Web surfing, and facilitating knowledge provenance tracking. In what follows, we present two Semantic Web based applications that rely on Swoogle.

VIII.D.1 Inference Web Search

Inference Web (IW) is a Semantic Web based framework for explaining reasoning tasks by storing, exchanging, combining, abstracting, annotating, comparing and rendering proofs and proof fragments provided by

reasoners [109]. The Inference Web ontology helps users to explicitly represent knowledge such as proofs and knowledge provenance information in PML (Proof Markup Language) documents. The PML document, therefore, enables users to cache proof so that (i) users may avoid regenerating the same proof by repeating time-consuming inference process; (ii) users may cache the snapshot of proof for archive purpose since the knowledge base may evolve over time; (iii) users may share proofs generated by different inference engines across multiple systems or applications.

In order to access PML documents on the Web, existing approaches are not effective. Manually book-making some PML documents that are used as entry point of a proof tree cannot offer information consumers an up-to-date and comprehensive index of online PML documents. It is also notable that Google only indexes several hundreds out of the over 10,000 online PML documents.

Therefore, Inference Web Search (IWSearch) is collaboratively developed by the Knowledge System Lab at Stanford and the eBiquity group at UMBC. IWSearch is a component of Inference Web infrastructure, and it is used to facilitate the Web aspects of Inference Web: it enables users to access all available PML documents on the Web. In order to maintain an up-to-date global view of the online PML documents, which can be updated, removed or added on the Web, the IWSearch component is needed. Two features are expected from IWSearch: (i) it should be able to find as many as possible PML documents on the Web, and (ii) it should be able to revisit online PML documents and actively maintain the corresponding metadata. Figure VIII.10 depicts the architecture of IWsearch.

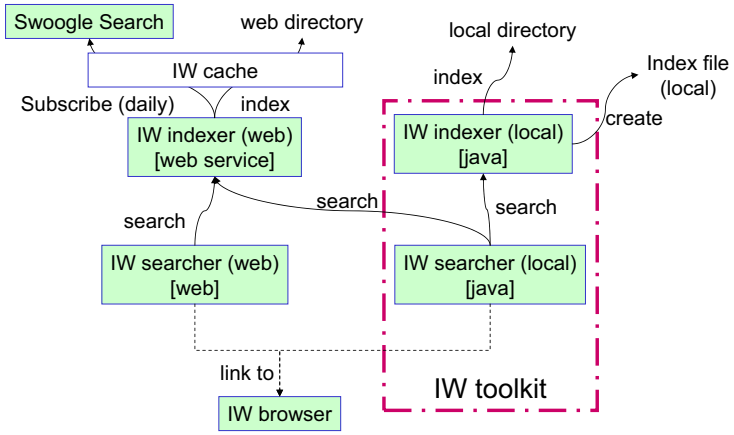


Figure VIII.10: The architecture of IWSearch

The *IW indexer* obtains PML documents from two sources: (i) it subscribes Swoogle Search service for newly discovered PML documents on the Web, and (ii) it traverses local file systems for PML documents.

The *IW indexer* also stores the metadata of index PML documents, such as the URL, the class-instances in a PML document, and the conclusions of *iw:NodeSet*. The *IW searcher* searches the metadata indexed by *IW indexer* and provides a search interface to the end-user. In particular, users may search instances of *iw:NodeSet* having the same conclusion as the one from a given justification tree to expand the justification tree with additional proofs. Since Inference Web applications are required to work in both *Web-mode* as a web service and in *local-mode* as a component of IW Toolkit, two implementations are built respectively.

IWSearch relies on Swoogle in harvesting PML documents on the Web. Swoogle have already indexed 53,412 PML documents from eight sources: five from Stanford university, one from W3C, one from 200.217.149.102 (a Uruguay IP address), and one from UIUC. Meanwhile, Swoogle's search interface also exposes PML documents to the entire Semantic Web research community and thus enhances the visibility of PML documents.

VIII.D.2 Detecting Conflict of Interests using Semantic Discovery

Conflict of Interest (COI) is well known as a situation where the decision could be biased by a variety of social relation factors such as family ties, business or friendship ties, and access to confidential information. Social relations are critical in detecting COIs; however, such information is not always available due to many reasons, such as privacy protection and the difficulties in aggregating and translating multiple social networks into the desired social networks.

In order to obtain and integrate multiple social networks from different sources on the Web, a system [3] has been built as part of the joint NSF project SEMDIS, which is conducted by researchers at University of Georgia and UMBC. Figure VIII.11 shows the solution architecture to the COI problem. The system brings together the DBLP data, which is stored in conventional database and published on the Web in well-formed structure, and the FOAF data, which is harvested from the Semantic Web in various partial structures. Once the desired social networks have been formed, the corresponding data is then stored in RDF database. Semantic association discovery mechanisms are then applied to find interesting paths. Once the hypotheses (i.e., the interesting patterns) have been found with certain degree of importance, the analysts may then check the sources of the hypotheses (in the form of an RDF graph) and then derive the trustworthiness of the hypotheses using the stored provenance information. Here, a hypothesis is measured by two metrics, the *importance* that is derived by semantic association and the *trustworthiness* that is derived by provenance based trust aggregation.

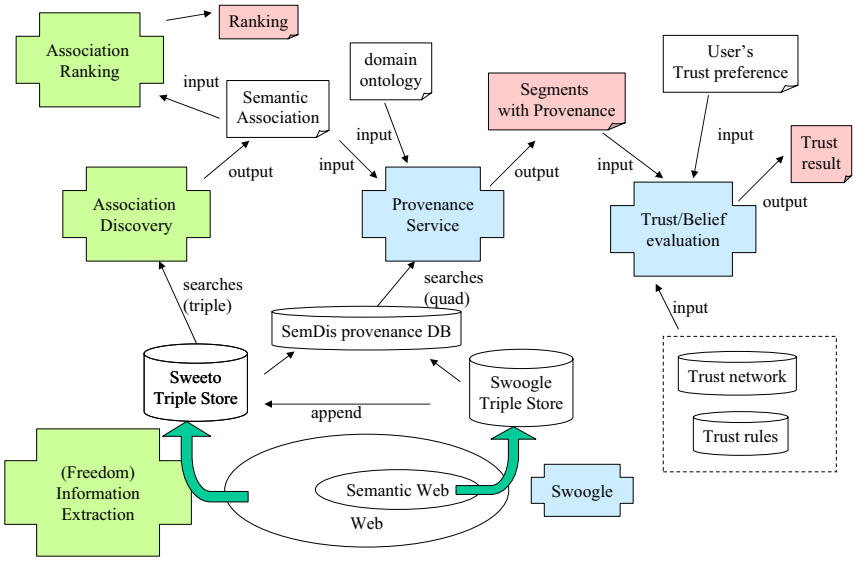


Figure VIII.11: The architecture of COI detection system

The COI detection system depends on two features described in this dissertation: (i) harvesting data from different sources such as DBLP and FOAF and (ii) integrating distributed data to form social networks and integrate multiple social networks into the desired social network.

Swoogle has indexed 512,973 FOAF documents from 105,544 websites[8]. Table VIII.5 lists top ten FOAF websites. We only choose a small portion of the dataset[9] to evaluate our system: a total of 49,750 FOAF documents have been chosen as the background knowledge and yield 207,413 instances of *foaf:Person*, and only 66,112 instances are kept because they have instantiated the property *foaf:name*.

Table VIII.5: Top 20 FOAF websites

website	#SWDs	website	#SWDs
http://www.tribe.net/	81034	http://www.fotothing.com/	6271
http://www.livejournal.com/	65391	http://www.crazylife.org/	6077
http://www.greatestjournal.com/	62779	http://blogs.dion.ne.jp/	6067
http://www.ecademy.com/	48128	http://d.hatena.ne.jp/.	5098
http://www.hackcraft.net/	16385	http://www.kwark.org/	5049
http://users.livejournal.com/	12875	http://www.blogware.com/	4189
http://blog.livedoor.jp/	11008	http://klab.lv/	3717
http://www.wasab.dk/	10355	http://www.mindswap.org/	3576
http://www.boards.ie/	9237	http://dannyayers.com/	3070
http://my.opera.com/	8146	http://elgg.net/	2902

[8]The result is produced by a pass of limited crawling, i.e., we retrieve no more than 10,000 Semantic Web documents from any website.

[9]The smaller dataset is obtained by heuristically selecting those FOAF documents that are relevant to education domain.

In this project, we separate the entity disambiguation process into three steps:

1. We first list the features about a person that can be used as identity:

   (a) **URI.** Any two named individuals sharing the same URI in RDF graph refer to the same person.

   (b) **InverseFunctionalProperty.** The FOAF ontology semantics defines unique identifiers of person, such as *foaf:mbox*, *foaf:mbox_sha1sum*, *foaf:homepage* and *foaf:weblog*, which are ideal clues to tell if two instances of *foaf:Person* refer to the same person.

   (c) **name.** Name could be a useful clue for identifying a person even though two different persons can share the same name. This clue is especially useful when combining with other clues, e.g., that both persons are from education domain.

   In our FOAF dataset, we have found 644 URIs, 11,405 mbox_sha1sums, 6,099 homepages, 3563 *weblogs*, and 757 *mboxs* being used as the identifiers of at least two instances of *foaf:person*.

2. We then create as many as possible relations between identified person entities without any merging action. For example, two entities are associated if they have the same value of email address, or they have the same last name. This approach saves the intermediate step of merging and lets us generate justification for future investigation.

3. Once the big graph has been created, several inference rules could be applied to derive the equivalent semantics between detected entities and conduct entity resolution.

Figure VIII.12 demonstrates the result of integrating Dr. Tim Finin's personal information from twelve sources. We can see two different values of *foaf:name* from two different sources in this case: (i) *Tim Finin* as stated in his FOAF profile and (ii) *Timothy W. Finin* as mentioned in `http://www-2.cs.cmu.edu/People/fgandon/foaf.rdf`. The latter is in fact the unique identifier of an author in DBLP[10]. However, a person described by FOAF documents may have two names because FOAF ontology does not require a person having unique name.

We should be careful in integrating information from multiple FOAF documents since some facts may be wrong and the merged facts may contain contradictions. Small errors in FOAF documents can lead to unexpected results. For example, some FOAF documents from blog.livedoor.jp, e.g., `http://blog.livedoor.jp/rusa95/foaf00756.rdf`, mistakenly assign the same *mbox_sha1sum* to different people. We have also found that *Jim Hendler* is wrongly fused with *Norman Walsh* by a FOAF document `http:`

[10] `http://www.informatik.uni-trier.de/~ley/db/`

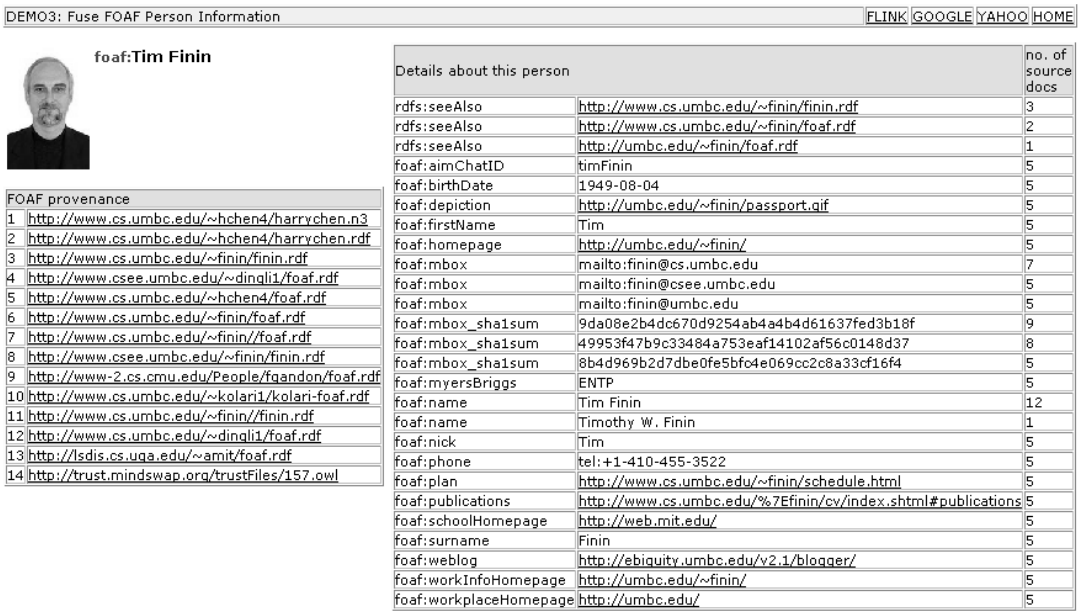


Figure VIII.12: Aggregating Tim Finin's person information from multiple sources

`//www.ilrt.bris.ac.uk/people/cmdjb/webwho.xrdf`, in which *foaf:mbox_sha1sum* was mistakenly associated with Norman's email-hash[11]. For this case, we need trust computation to resolve the conflict.

VIII.E Summary

In this chapter, we describe several useful Semantic Web based applications that depend on the work presented in previous chapters. The importance and usefulness of our work is justified by positive feedback from users as well as the importance of these applications.

[11] Norman's email-hash is *ef99fd659575b85b94575cc016043813ec1294dc* according to `http://norman.walsh.name/knows/who#norman-walsh`

Chapter IX

CONCLUSIONS AND FUTURE DIRECTIONS

The Semantic Web, from the Web aspect, is not merely one universal RDF graph; instead, it is a *web of data* that is distributed on the Web and a *web of belief* that hosts beliefs published by independent agents. In this dissertation, we present a systematic approach, including conceptual models and enabling tools, to address the Web aspect of the Semantic Web and facilitate the emergence of the Semantic Web on the Web.

IX.A Meta-description of the Semantic Web

In this dissertation, the *Web Of Belief* ontology is introduced to capture the content and the context of Semantic Web data and enable explicit meta-description of the Semantic Web on the Web. The WOB ontology is not *yet another ontology* because it significantly enhances the current model of Semantic Web by acknowledging the Web aspect of the Semantic Web.

The WOB ontology is designed as a core ontology that covers the most important and general concepts and relations for describing the Semantic Web:

- Semantic Web document
- Semantic Web Term
- RDF graph reference
- agent and person
- provenance, such as where, whom, why and definition

- modal assertions such as belief and trust

The WOB ontology is also highly extensible. It can be associated with many existing ontologies such as Dublin Core, OWL ontology and PML ontology.

Our future work may extend the WOB ontology to capture specific concepts in modeling the Semantic Web. For example, we may extend *wob:GraphReferenc* and implement corresponding tools to realize the freedom of citing any RDF graph on the Semantic Web, especially those giant RDF graphs stored in RDF database. We may also extend the modal assertions by incorporating various existing trust and belief representation schema to realize the actual vision of *web of belief*.

IX.B Global catalog and property of the Semantic Web

A global catalog of the Semantic Web on the Web has been obtained and is actively maintained by a hybrid harvesting framework, which integrates three existing automated harvesting methods, namely Google based meta-crawling, bounded HTML crawling, and RDF crawling. The hybrid approach is highlighted by automating the harvesting process: (i) manual submission feeds seeding URLs to the *Google based meta-crawler* and the *bounded HTML crawler*; (ii) the meta-crawler and the HTML crawler automatically find and then feed seeding URLs to the *RDF crawler*, and (iii) the harvesting results of the RDF crawler can be used to inductively derive new seeding URLs for meta-crawler and HTML crawler.

The framework is effective because it has harvested significant amount of Semantic Web documents (over 1.2 million Semantic Web documents including over 10,000 Semantic Web ontologies) and has achieved high harvesting accuracy (over 42% harvested URLs are confirmed SWDs).

The global properties of the Semantic Web are measured using Google's estimation and the sample dataset *SW06MAR* collected by the hybrid harvesting framework. Based on Google, we estimate that the number of indexable Semantic Web documents is between 10^7 and 10^9. We also show that the sample dataset *SW06MAR* is significant enough for deriving global properties. It is notable that (invariant) Power distribution is frequently observed in different statistics about the Semantic Web, such as the number of SWDs per website, the age of SWD, the number of triples being used to define SWT, and the number of class-instances of SWT. All these observations support the hypothesis that the Semantic Web is an emerging *complex system*.

Our future work may improve the performance of hybrid harvesting method with enhanced heuristics and efficient implementation.

IX.C Surfing the Semantic Web

Based on the WOB ontology, we model Web-scale Semantic Web data access behaviors using two conceptual models, namely the *current Semantic Web search and navigation model*, which reveals the limitations in Semantic Web surfing based on existing technologies, and the *enhanced Semantic Web search and navigation model*, which improves the current model by adding new Semantic Web search services and enriching navigation network. We also develop Swoogle, a Semantic Web search engine, to enable the *enhanced model*. Swoogle is highlighted by its rich metadata of the Semantic Web. The *current model* leads to a rational surfing algorithm that ranks Semantic Web documents and terms by popularity. The ranking algorithm is highlighted by its explainable ranking heuristics and results.

In order to support tracking knowledge provenance, we propose the concept *RDF molecule* that preserves both the extension and the binding semantics of *blank node*. The *RDF molecule* is especially useful in finding partial supporting evidence of a given RDF graph on the Web.

One set of open problems involves scalability issues. Techniques that work today with 10^6 Semantic Web documents may fail when the Semantic Web has 10^8 documents. When we have indexed 10^8 Semantic Web documents, will Swoogle's metadata still be manageable? Google has successfully indexed 8 billion of Web document; however, the metadata of conventional Web documents is much simpler than the metadata of Semantic Web data. In order to track knowledge provenance, users need effective methods to retrieve the supporting evidence of a given RDF graph. This cannot be done by storing and searching all the Semantic Web documents in one RDF storage system because such storage violates the freedom of distributed publishing. Therefore, we need investigate effective methods in the future to federate the knowledge provenance services maintained by independent knowledge providers.

IX.D Semantic Web Applications

We evaluate our work in some Semantic Web applications, including the enabling applications and the Semantic Web based applications.

The enabling applications include Swoogle APIs, Swoogle website, Swoogle archive service and Swoogle Ontology Dictionary. These applications support Semantic Web data access using the harvested global catalog and demonstrate the important features of our conceptual models and enabling tools. Through the statistics of their access log, such as web page hits and the number of unique query strings, we observe positive

evidences showing the great interest from the Semantic Web community.

The Semantic Web based applications, which usually use Semantic Web as a database or knowledge base, need our enabling tools to access Semantic Web data on the Web. These applications, such as *Inference Web Search* and *COI detection system* are by themselves important in addressing real world problems. Their importance and success help us justify the applicability of the Semantic Web on the Web as well as the utility of our conceptual models and enabling tools.

Our future work may enhance the work mentioned in this dissertation. For example, we can enhance Swoogle search interface by helping users generate the FROM clause of SPARQL query. We can also enhance Swoogle Ontology Dictionary by (i) adding more ranking components such as trust based ranking and (ii) offering an effective implementation for the ontology DIY mechanism.

Our future may also develop additional useful applications, e.g. to build a domain knowledge base. A domain knowledge base hosts Semantic Web data within a domain. The data is actively maintained by subscribing Swoogle's daily report (in the form of an RSS) on the changes of SWDs relevant to the subscribed Swoogle queries. While Swoogle search is free of inference, a domain knowledge base allows users to specify inference support, e.g. RDFS inference and OWL Lite inference. Such domain knowledge base can be used to expose the inference capability of the Semantic Web within a data space bounded by the specified domain. Moreover, the implementation for storing RDF data may be optimized for the domain data. For example, a domain knowledge base for FOAF need efficient design that supports special treatment on the *owl:InverseFunctionalProperty*, and a domain knowledge base for Wordnet need efficient design that supports the deep class hierarchy.

Appendix A

ABBREVIATIONS

Prefixes and corresponding namespaces

prefix	namesapce URI
cc	http://web.resource.org/cc/
daml	http://www.daml.org/2001/03/daml+oil#
dc	http://purl.org/dc/elements/1.1/
dcterms	http://purl.org/dc/terms/
foaf	http://xmlns.com/foaf/0.1/
geo	http://www.w3.org/2003/01/geo/wgs84_pos#
mcvb	http://webns.net/mvcb/
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
rdftest	http://www.w3.org/2000/10/rdf-tests/rdfcore/testSchema
rss	http://purl.org/rss/1.0/
swoogle	http://daml.umbc.edu/ontologies/webofbelief/1.4/swoogle.owl#
wn	http://xmlns.com/wordnet/1.6/
wob	http://daml.umbc.edu/ontologies/webofbelief/1.4/wob.owl#
wobGraph	http://daml.umbc.edu/ontologies/webofbelief/1.4/wob-ext-graph.owl

Acronyms

DAML – DARPA Agent Markup Language

FOAF – Friend-Of-A-Friend ontology [20]

HTTP Hypertext Transfer Protocol [53]

IFP – owl:InverseFunctionalProperty

OWL – Web Ontology Language [35]

RDF – Resource Description Framework [95]

RDFS – RDF Schema [79]

RSS – RDF Site Summary Ontology [130]

SVG – Scalable Vector Graphics, see http://www.w3.org/Graphics/SVG/.

BIBLIOGRAPHY

[1] H. Alani and C. Brewster. Ontology ranking based on the analysis of concept structures. In *Proceedings of the 3rd International Conference on Knowledge Capture (K-Cap)*, 2005.

[2] B. Aleman-Meza, C. Halaschek, A. Sheth, I. B. Arpinar, and G. Sannapareddy. SWETO: Large-Scale Semantic Web Test-bed. In *Proceedings of the 16th International Conference on Software Engineering and Knowledge Enginerring (SEKE2004): Workshop on Ontology in Action*, 2004.

[3] B. Aleman-Meza, M. Nagarajan, C. Ramakrishnan, A. Sheth, B. Arpinar, L. Ding, P. Kolari, A. Joshi, and T. Finin. Semantic analytics on social networks: Experiences in addressing the problem of conflict of interest detection. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, 2006.

[4] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The RDFSuite: Managing Voluminous RDF Description Bases. In *Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb'01) in conjunction with WWW10*, May 2001.

[5] K. Apsitis, S. Staab, S. Handschuh, and H. Oppermann. Specification of an RDF Crawler. `http://ontobroker.semanticweb.org/rdfcrawl/help/specification.html` (last visited on March 2006), 2005.

[6] A. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311(3-4):590–614, 2002.

[7] D. Beckett. Scalability and Storage: Survey of Free Software/Open Source RDF storage systems. Technical Report 1016, ILRT, 2002.

[8] D. Beckett. RDF/XML syntax specification (revised). http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/, February 2004.

[9] T. Berners-Lee. Notation 3 – Ideas about Web architecture. http://www.w3.org/DesignIssues/Notation3.html (last visited on March 2006), 1998.

[10] T. Berners-Lee. Semantic web road map. http://www.w3.org/DesignIssues/Semantic.html (last visited on March 2006), 1998.

[11] T. Berners-Lee and D. Connolly. Delta: an ontology for the distribution of differences between rdf graphs. http://www.w3.org/DesignIssues/Diff (last visited on March 2006), 2004.

[12] T. Berners-Lee, R. Fielding, and L. Masinter. Rfc 2396 - uniform resource identifiers (uri): Generic syntax. http://www.faqs.org/rfcs/rfc2396.html (last visited on March 2006), 1998.

[13] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):35–43, 2001.

[14] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[15] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the 7th international conference on World Wide Web*, pages 379–388, 1998.

[16] M. Biddulph. Crawling the semantic web. In *XML Europe*, 2004.

[17] C. Bizer and A. Seaborne. D2RQ Treating Non-RDF Databases as Virtual RDF Graphs. In *ISWC'04: Proceedings of the 3rd International Semantic Web Conference*, 2004.

[18] D. Brickley. Scutter spec. http://rdfweb.org/topic/ScutterSpec (last visited on March 2006), 2003.

[19] D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF schema. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, February 2004.

[20] D. Brickley and L. Miller. FOAF Vocabulary Specification. *http://xmlns.com/foaf/0.1/* (last visited on March 2006), 2004.

[21] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An Architecture for Storing and Querying RDF and RDF Schema. In *ISWC '02: Proceedings of the First International Semantic Web Conference*, pages 54–68, 2002.

[22] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *International Conference o Database Theory (ICDT)*, pages 316–330, 2001.

[23] M. Cai and M. Frank. Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 650–657, 2004.

[24] J. J. Carroll. Matching RDF graphs. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 5–15, 2002.

[25] J. J. Carroll. Signing RDF graphs. Technical Report HPL-2003-142, HP Lab, Jul 2003.

[26] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. Technical Report HPL-2004-57, HP Lab, May 2004.

[27] J. J. Carroll and J. D. Roo. OWL web ontology language test cases. `http://www.w3.org/TR/2004/REC-owl-test-20040210/`, February 2004.

[28] V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis. On labeling schemes for the semantic web. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 544–555, 2003.

[29] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. on Database Systems*, 25(2):179–227, June 2000.

[30] P. P. da Silva, D. L. McGuinness, and R. Fikes. A proof markup language for semantic web services. Technical Report KSL-04-01, Stanford, 2004.

[31] P. P. da Silva, D. L. McGuinness, and R. McCool. Knowledge provenance infrastructure. *Data Engineering Bulletin*, 26(4):26–32, 2003.

[32] J. Davies, D. Fensel, and F. van Harmelen, editors. *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley & Sons, 2003.

[33] J. Davies, R. Weeks, and U. Krohn. Quizrdf: search technology for thesemantic web. In *workshop on RDF and Semantic Web Applications in 11th International Conference on World Wide Web*, 2002.

[34] M. Dean and K. Barber. Daml crawler. `http://www.daml.org/crawler/` (last visited on March 2006), 2002.

[35] M. Dean and G. Schreiber. OWL web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, February 2004.

[36] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In *Proceedings of the IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics- Semantic Issues in Multimedia Systems*, pages 351–369, 1999.

[37] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the web. In F. Baader and U. Sattler, editors, *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*, 2000.

[38] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: The roles of XML and RDF. *IEEE Internet Computing, special issue "Knowledge Networking"*, 4(5):63–74, Sept./Oct. 2000.

[39] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. SemTag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW' 03: Proceedings of the Twelfth International World Wide Web Conference*, pages 178–186, 2003.

[40] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs. Swoogle: A search and metadata engine for the semantic web. In *CIKM'04: the Proceedings of ACM Thirteenth Conference on Information and Knowledge Management*, 2004.

[41] L. Ding, T. Finin, Y. Peng, P. P. da Silva, and D. L. McGuinness. Tracking rdf graph provenance using rdf molecules. Technical Report TR-05-06, UMBC, 2005.

[42] L. Ding, P. Kolari, S. Ganjugunte, T. Finin, and A. Joshi. Modeling and evaluating trust network inference. In *Proceedings of the Seventh International Workshop on Trust in Agent Societies at AAMAS'2004*, 2004.

[43] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and ranking knowledge on the semantic webb. In *ISWC'05: proceedings of the 4th International Semantic Web Conference*, November 2005.

[44] L. Ding, L. Zhou, T. Finin, and A. Joshi. How the semantic web is being used:an analysis of foaf. In *Proceedings of the 38th International Conference on System Sciences*, January 2005.

[45] Y. Ding and D. Fensel. Ontology library systems: The key to successful ontology reuse. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, pages 93–112, 2001.

[46] Dublin Core Collection Description Working Group. Dublin core collection description proposed term : Provenance. `http://www.ukoln.ac.uk/metadata/dcmi/collection-provenance/` (last visited on March 2006), 2004.

[47] M. Dzbor, J. Domingue, and E. Motta. Magpie - towards a semantic web browser. In *ISWC'02: Proceedings of the Second International Semantic Web Conference*, pages 690–705, 2003.

[48] A. Eberhart. Survey of rdf data on the web. Technical report, International University in Germany, 2002.

[49] M. Erdmann, A. Maedche, H. Schnurr, and S. Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In *Proceedings of the COLING-2000 Workshop on Semantic Annotation and Intelligent Content*, 2000.

[50] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.

[51] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 1969.

[52] D. Fensel, C. Bussler, Y. Ding, V. Kartseva, M. Klein, M. Korotkiy, B. Omelayenko, and R. Siebes. Semantic web application areas. In *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002)*, 2002.

[53] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical Report Internet RFC 2616, IETF, 1998. `http://www.ietf.org/rfc/rfc2616.txt`.

[54] M. Fox and J. Huang. Knowledge provenance: An approach to modeling and maintaining the evolution and validity of knowledge. Technical report, University of Toronto, 2003.

[55] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. A theoretical framework for ontology evaluation and validation. In *Proceedings of the 2nd Italian Semantic Web Workshop (SWAP 2005)*, 2005.

[56] R. Gil, R. Garca, and J. Delgado. Measuring the semantic web. *AIS SIGSEMIS Bulletin*, 1(2):69–72, June 2004.

[57] J. Golbeck, B. Parsia, and J. Hendler. Trust networks on the semantic web. In *Proceedings of Cooperative Intelligent Agents*, 2003.

[58] J. Grant and D. Beckett. RDF test cases. `http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/`, February 2004.

[59] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Automatic partitioning of owl ontologies using e-connections. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), January 2005.

[60] G. A. Grimnes, P. Edwards, and A. Preece. Learning meta-descriptions of the foaf network. In *ISWC'04: Proceedings of the 3rd International Semantic Web Conference*, 2004.

[61] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.

[62] R. Guha. Open rating systems. In *Proceedings of 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, 2004.

[63] R. Guha and R. McCool. TAP: a Semantic Web platform. *Computer Networks*, 42(5):557–577, 2003.

[64] R. Guha, R. McCool, and E. Miller. Semantic search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 700–709, 2003.

[65] R. V. Guha. Object co-identification. In *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, 2004.

[66] R. V. Guha, R. McCool, and R. Fikes. Contexts for the semantic web. In *ISWC'04: Proceedings of the 3rd International Semantic Web Conference*, 2004.

[67] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW'05: The 14th International World Wide Web Conference (poster paper)*, 2005.

[68] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large owl datasets. In *ISWC'04: Proceedings of the 3rd International Semantic Web Conference*, pages 274–288, 2004.

[69] C. Gutierrez, C. Hurtado, and A. O. Mendelzon. Foundations of semantic web databases. In *PODS '04: Proceedings of the 23rd ACM symposium on principles of database systems*, pages 95–106, 2004.

[70] R. Guttman, A. Moukas, and P. Maes. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review*, 13(2):147–159, 1998.

[71] P. Haase, J. Broekstra, A. Eberhart, and R. Volz. A comparison of rdf query languages. In *Proceedings of the 3rd International Semantic Web Conference*, Nov 2004.

[72] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *WWW'03: Proceedings of the 12th International Conference on World Wide Web*, pages 556–567, 2003.

[73] S. Handschuh and S. Staab. Authoring and annotation of web pages in cream. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 462–473, 2002.

[74] S. Handschuh and S. Staab. *Annotation for the Semantic Web*. IOS Press, 2003.

[75] S. Handschuh and S. Staab. Cream: Creating metadata for the semantic web. *Comput. Networks*, 42(5):579–598, 2003.

[76] S. Harris and N. Gibbins. 3store: Efficient bulk RDF storage. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03)*, pages 1–20, 2003. `http://eprints.aktors.org/archive/00000273/`.

[77] A. Harth. An Integration Site for Semantic Web Metadata. *Journal of Web Semantics*, 1(2):299–234, 2004.

[78] J. Hartmann, Y. Sure, A. Giboin, D. Maynard, M. del Carmen Surez-Figueroa, and R. Cuel. Methods for ontology evaluation. Technical report, University of Karlsruhe, DEC 2004.

[79] P. Hayes. RDF semantics. `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, February 2004.

[80] T. Heath, M. Dzbor, and E. Motta. Supporting user tasks and context: Challenges for semantic web research. In *Proceedings of the Workshop on End-user Aspects of the Semantic Web (UserSWeb) in conjunction with European Semantic Web Conference (ESWC2005)*, 2005.

[81] I. Horrocks. DAML+OIL: A description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9, 2002.

[82] I. Horrocks and S. Tessaris. Querying the semantic web: A formal approach. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 177–191, 2002.

[83] DAML Ontology Library. `http://www.daml.org/ontologies/` (last visited on March 2006).

[84] Ontaria. `http://www.w3.org/2004/ontaria/`(offline based on last visited on March 2006).

[85] Schema Web . `http://www.schemaweb.info/`(last visited on March 2006).

[86] Semantic web search. `http://www.semanticwebsearch.com/`(last visited on March 2006).

[87] N. R. Jennings, P. Faratin, M. J. Johnson, P. O'Brien, and M. E. Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 5(2,3):105– 130, 1996.

[88] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[89] John C. Paolillo and Elijah Wright. The Challenges of FOAF Characterization. In *Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the (Semantic) Web*, 2004.

[90] J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared web annotations. In *Proceedings of the 10th International World Wide Web Conference*, pages 623–632, 2001.

[91] A. Kalyanpur, B. Parsia, and J. Hendler. A tool for working with web ontologies. *International Journal on Semantic Web and Information Systems*, 1(1), 2005.

[92] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A declarative query language for RDF. In *WWW'02: Proceedings of the 11th International Conference on World Wide Web*, 2002.

[93] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology versioning and change detection on the web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, 2002.

[94] G. Klyne. Circumstance, provenance and partial knowledge. `http://www.ninebynine.org/RDFNotes/UsingContextsWithRDF.html`(last visited on March 2006), 2002.

[95] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, February 2004.

[96] P. Kogut and W. Holmes. Aerodaml: Applying information extraction to generate daml annotations from web pages. In *Workshop Knowledge Markup and Semantic Annotation in K-CAP 2001*, 2001.

[97] M.-R. Koivunen and E. Miller. W3c semantic web activity. In *Semantic Web Kick-Off in Finland: Vision, Technologies,Research, and Applications*, 2002.

[98] O. Kutz. *E-connections and logics of distance*. PhD thesis, University of Liverpool, 2004.

[99] O. Kutz, C. Lutz, F.Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[100] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[101] R. Lee. Scalability report on triple store applications. Technical report, MIT, 2004.

[102] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd. Cyc: toward programs with common sense. *Commun. ACM*, 33(8):30–49, 1990.

[103] A. Lozano-Tello and A. Gomez-Perez. Ontometric:a method to choose the appropriate ontology. *Journal of Database Management*, 15(2):1–18, 2003.

[104] S. Lu, M. Dong, and F. Fotouhi. The semantic web: Opportunities and challenges for next-generation web applications. *Information Research*, 7(4), 2002.

[105] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based web agents. In *Proceedings of the First International Conference on Autonomous Agents (Agents97)*, pages 59–66, 1997.

[106] A. Maedche and S. Staab. KAON: The karlsruhe ontology and semantic web meta project. *Künstliche Intelligenz*, 17(3):27–30, 2003.

[107] A. Magkanaraki, S. Alexaki, V. Christophides, and D. Plexousakis. Benchmarking rdf schemas for the semantic web. In *ISWC'02: Proceedings of the First International Semantic Web Conference*, pages 132–146, 2002.

[108] P. Martin and P. Eklund. Embedding knowledge in web documents. In *WWW'99: Proceedings of the 8th International Conference on World Wide Web*, pages 324–341, 1999.

[109] D. L. McGuinness and P. P. da Silva. Explaining answers from the semantic web: the inference web approach. *Journal of Web Semantics*, 1(4):397–413, 2004.

[110] P. Mika. Bootstrapping the FOAF-Web: An Experiment in Social Network Mining. In *Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, 2004.

[111] P. Mika. Social Networks and the Semantic Web: An Experiment in Online Social Network Analysis. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2004.

[112] P. Mika. Flink: Semantic web technology for the extraction and analysis of social networks. *Journal of Web Semantics*, 3(2), 2005.

[113] M. Minsky. A framework for representing knowledge. Technical report, MIT, 1974.

[114] K. Möller. GECO - Using Human Language Technology to Enhance Semantic Web Browsing. In *Proceedings of Faculty of Engineering Research Day 2004, National University of Ireland, Galway*, April 2004.

[115] K. Mollerand and L. P. andDaniel Bachlechner. Portal ontology. `http://sw-portal.deri.at/papers/deliverables/D1_v1.2PortalOntology.pdf`, 2004.

[116] W. Nejdl, W. Siberski, and M. Sintek. Design issues and challenges for rdf- and schema-based peer-to-peer systems. *SIGMOD Rec.*, 32(3):41–46, 2003.

[117] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: a P2P networking infrastructure based on RDF. In *WWW'02: Proceedings of the 11th International Conference on World Wide Web*, pages 604–615, 2002.

[118] N. Noy and M. Musen. Evaluating ontology-mapping tools: requirements and experience. In *Proc. 1st workshop on Evaluation of Ontology Tools (EON2002), EKAW'02*, 2002.

[119] N. F. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, July 2004.

[120] N. F. Noy, S. Kunnatur, M. Klein, and M. A. Musen. Tracking changes during ontology evolution. In *ISWC'04: Proceedings of the 3rd Internatonal Semantic Web Conference*, Nov., 7 – 11 2004.

[121] B. C. Ooi, K.-L. Tan, A. Zhou, C. H. Goh, Y. Li, C. Y. Liau, B. Ling, W. S. Ng, Y. Shu, X. Wang, and M. Zhang. PeerDB: peering into personal databases. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 659–659, 2003.

[122] A. Owens. Semantic storage: Overview and assessment. Technical Report 11985, University of Southampton, 2005.

[123] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[124] S. B. Palmer. RDF in HTML: Approaches. `http://infomesh.net/2002/rdfinhtml/`, 2002. `http://infomesh.net/2002/rdfinhtml/`.

[125] Z. Pan. Benchmarking dl reasoners using realistic ontologies. In *Proceedings of the International workshop on OWL: Experience and Directions*, 2005.

[126] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging owl ontologies. In *WWW'05: Proceedings of the 14th International Conference on World Wide Web*, May 2005.

[127] C. Patel, K. Supekar, Y. Lee, and E. K. Park. OntoKhoj: a semantic web portal for ontology searching, ranking and classification. In *WIDM'03: Proceedings of the 5th ACM international workshop on Web information and data management*, pages 58–61, 2003.

[128] Y. Peng and J. Reggia. *Abductive Inference Models for Diagnostic Problem Solving*. Springer-Verlag, 1990.

[129] F. Perich. *On Peer-to-Peer Data Management in Pervasive Computing Environments*. PhD thesis, UMBC, May 2004.

[130] M. Pilgrim. What is RSS? `http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html` (last visited on March 2006), 2002.

[131] J. E. Pitkow. Summary of www characterizations. *Computer Networks*, 30(1-7):551–558, 1998.

[132] B. Popov, A. Kiryakov, D. Manov, D. Ognyanoff, and M. Goranov. KIM - Semantic Annotation Platform. In *ISWC'03: Proceedings of the Second International Semantic Web Conference*, pages 834–849, 2003.

[133] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. `http://www.w3.org/TR/2006/WD-rdf-sparql-query-20060220/`, 2006.

[134] D. A. Quan and R. Karger. How to make a semantic web browser. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 255–265, 2004.

[135] L. Reeve and H. Han. Survey of semantic annotation platforms. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1634–1638, 2005.

[136] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *ISWC'03: Proceedings of the Second International Semantic Web Conference*, 2003.

[137] C. J. V. Rijsbergen. *Information Retrieval(2nd edition)*. Dept. of Computer Science, University of Glasgow, 1979.

[138] C. Rocha, D. Schwabe, and M. P. Aragao. A hybrid approach for searching in the semantic web. In *WWW'04: Proceedings of the 13th international conference on World Wide Web*, pages 374–383, 2004.

[139] A. Seaborne. RDQL - A Query Language for RDF (w3c member submission 9 january 2004). `http://www.w3.org/Submission/RDQL/`, 2004.

[140] N. Shadbolt. A matter of trust. *IEEE Intelligent Systems*, 17(1):2–3, 2002.

[141] U. Shah, T. Finin, and A. Joshi. Information retrieval on the semantic web. In *Proceedings of the 11th international conference on Information and knowledge management*, pages 461–468, 2002.

[142] C. Sherman. Meta search engines: An introduction. `http://searchenginewatch.com/searchday/article.php/2160771` (last visited on March 2006), September 2002.

[143] C. Sherman. Metacrawlers and metasearch engines. `http://searchenginewatch.com/links/article.php/2156241` (last visited on March 2006), March 2004.

[144] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke. Managing semantic content for the web. *IEEE Internet Computing*, 6(4):80–87, 2002.

[145] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[146] M. Sintek and S. Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In *ISWC'01: Proceedings of the 1st International Semantic Web Conference*, 2002.

[147] P. Stickler. CBD - Concise Bounded Description (w3c member submission 30 september 2004). `http://www.w3.org/Submission/2004/SUBM-CBD-20040930/`, 2004.

[148] H. Stuckenschmidt and M. C. A. Klein. Structure-based partitioning of large concept hierarchies. In *ISWC'04: Proceedings of the 3rd International Semantic Web Conference*, pages 289–303, 2004.

[149] K. Supekar, C. Patel, and Y. Lee. Characterizing quality of knowledge on semantic web. In *Proc. of 7th International Florida Artificial Intelligence Research Society Conf.*, 2002.

[150] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. Ontoqa: Metric-based ontology quality analysis. In *Proceedings of IEEE ICDM 2005 Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources.*, 2006.

[151] C. Tempich and R. Volz. Towards a benchmark for semantic web reasoners - an analysis of the daml ontology library. In Y. Sure, editor, *Proceedings of Evaluation of Ontology-based Tools (EON2003) at 2nd International Semantic Web Conference (ISWC 2003)*, pages 4–15, OCT 2003.

[152] G. Tummarello, C. Morbidoni, J. Petersson, P. Puliti, and F. Piazza. RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications. In *International Workshop on Peer-to-Peer Knowledge Management (P2PKM)*, 2004.

[153] G. Tummarello, C. Morbidoni, P. Puliti, and F. Piazza. Signing individual fragments of an rdf graph. In *WWW (Special interest tracks and posters) 2005*, pages 1020–1021, 2005.

[154] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of the IGPL*, 11(2):133–157, 2003.

[155] R. Volz, D. Oberle, and A. Maedche. Towards a modularized semantic web. In *Proceedings of the ECAI'02 Workshop on Ontologies and Semantic Interoperability*, 2002.

[156] E. W. Weisstein. Order of magnitude. `http://mathworld.wolfram.com/OrderofMagnitude.html` (last visited on March 2006).

[157] C. A. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data Knowledge Engineering*, 39(1):51–74, 2001.

[158] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds. Efficient RDF Storage and Retrieval in Jena2. In *Proceedings of SWDB'03, 1st International Workshop on Semantic Web and Databases, Co-located with VLDB 2003*, pages 131–150, Sept. 2003.

[159] D. Wood, P. Gearon, and T. Adams. Kowari: A platform for semantic web storage and analysis. In *Proceedings of XTech 2005*, 2005.

[160] G. Yang and M. Kifer. Reasoning about anonymous resources and meta statements on the semantic web. *Journal on Data Semantics*, 1:69–97, 2003.

[161] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, 1999.

[162] H. Yao, A. M. Orme, and L. Etzkorn. Cohesion metrics for ontology design and application. *Journal of Computer Science*, 1(1):107–113, 2005.

[163] L. Zhang, Y. Yu, J. Zhou, C. Lin, and Y. Yang. An enhanced model for searching in semantic portals. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 453–462, 2005.

[164] Y. Zhang, W. Vasconcelos, and D. Sleeman. Ontosearch: An ontology search engine. In *The Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2004.

Index