16 Semantic Web Search Engines

Mathieu d'Aquin[1] · *Li Ding*[2] · *Enrico Motta*[1]
[1]The Open University, Milton Keynes, UK
[2]Rensselaer Polytechnic Institute, Troy, NY, USA

John Domingue, Dieter Fensel & James A. Hendler (eds.), *Handbook of Semantic Web Technologies*,
DOI 10.1007/978-3-540-92913-0_16,

Abstract: The last couple of years have seen an increasing growth in the amount of Semantic Web data made available, and exploitable, on the Web. Compared to the Web, one unique feature of the Semantic Web is its friendly interface with software programs. In order to better serve human users with software programs, supporting infrastructures for finding and selecting the distributed online Semantic Web data are needed. A number of Semantic Web search engines have emerged recently. These systems are based on different design principles and provide different levels of support for users and/or applications. In this chapter, a survey of these Semantic Web search engines is presented, together with the detailed description of the design of two prominent systems: Swoogle and Watson. The way these systems are used to enable domain applications and support cutting-edge research on Semantic Web technologies is also discussed. In particular, this chapter includes examples of a new generation of semantic applications that, thanks to Semantic Web search engines, exploit online knowledge at runtime, without the need for laborious acquisition in specific domains. In addition, through collecting large amounts of semantic content online, Semantic Web search engines such as Watson and Swoogle allow researchers to better understand how knowledge is formally published online and how Semantic Web technologies are used. In other terms, by mining the collected semantic documents, it becomes possible to get an overview and explore the Semantic Web landscape today.

The first section below (Sect. 16.1) presents a general overview of the area, including the main challenges, related systems, as well as an abstract specification of what is called Semantic Web search engines. It also includes a detailed overview of the two systems more specifically considered as case studies, Swoogle (Sect. 16.1.4) and Watson (Sect. 16.1.5). Section 16.2 shows how these systems are currently being used and applied, both as development platforms to make possible the realization of applications exploiting Semantic Web content (Sect. 16.2.1), and as research platforms, allowing one to better understand the content of the Semantic Web, how knowledge is published online and how it is structured. Finally, Sect. 16.3 briefly introduces other resources to be considered in the area of Semantic Web search engines, and Sect. 16.4 concludes the chapter.

16.1 Scientific and Technical Overview

In the early years, the deployment of the Semantic Web has been hindered by a dilemma on ontology reuse: ontology developers wanted others to adopt ontologies they created but they seldom adopted the ontologies created by others. Ontologies and knowledge bases were generally tailored to fit specific domain applications, which were rarely open to multiple, external ontologies and did not have to tackle the issues related to data integration, ontology coevolution, etc. This situation could be attributed to a number of reasons such as the existence of alternative standards, formalisms and languages (e.g., RDF and Conceptual Graphs [1]), the difficulties in integrating knowledge from different sources (e.g., DAML time ontology [2] and SOUPA [3] time ontology), and most importantly, to the limited support for finding and selecting reusable knowledge on the Web.

With the great efforts on standardization (see, e.g., RDF [4], OWL [5], URIs for the Semantic Web [6], SPARQL [7]), the fast-growing linked data (see, e.g., [8, 9]), and the advance of technologies such as robust storage, querying, and manipulation systems, the Semantic Web is now deemed as a huge success, at least according to one particular measure – availability: vast amounts of Semantic Web data are now directly made accessible from the Web for applications to reuse; SPARQL endpoints have been deployed all over the world to host particular datasets in specific domains; and more and more datasets encoded in relatively confined ontologies are now getting linked to the linked data cloud, which is leading to the ultimate "Web of Data" vision of the Semantic Web.

As a consequence, new challenges emerge surrounding the data accessibility issues: How to make the huge amount of Semantic Web data and data services published on the Web accessible by Web users, especially those unexpected consumers who are not familiar with the published datasets? How to facilitate applications access and integrate distributed Semantic Web data at web-scale? What kind of applications and research can be conducted with access to all the Semantic Web data published on the Web? What sort of support is needed by these applications for effectively using such knowledge? Semantic Web search engines, therefore, are developed to address these issues.

16.1.1 Challenges

The core challenges surrounding data accessibility can be summarized as making ontologies and data distributed on the Web accessible by intelligent applications to effectively take advantage of the Semantic Web as a distributed and interlinked knowledge base. Of course, more specific challenges emerge from this goal:

Heterogeneity: Despite the effort in standardizing technologies, at a higher level, the Semantic Web is characterized by heterogeneity along several dimensions, such as ontology quality, complexity, modeling, and views. A nontrivial effort is necessary to provide a homogeneous view and homogeneous access mechanisms to such heterogeneous information.

Scalability: With its millions of documents and billions of triples, the Semantic Web is already well beyond the size of any existing knowledge base in any semantic application. Although applications and users of the Semantic Web typically focus on a subset of what is available, efficient access mechanisms are required, and a shift is necessary for applications to locate and process the relevant information. Moreover, the open nature and the current rate of growth of the Semantic Web make it unrealistic to keep all Semantic Web data in a completely centralized manner; therefore, it is always desired to have relevant Semantic Web documents filtered before use.

Quality: Perfect quality cannot be assumed even in the absence of parser failure or semantic inconsistency. Information on the Semantic Web originates from many different sources and therefore varies considerably in quality. Trust becomes a key factor in using the Semantic Web and increasing amount of interests have been projected on ranking both the importance of Semantic Web resources, and the level of confidence with which these resources can be used.

16.1.2 Related Systems

Several Semantic Web search engines have recently appeared (see ➔ Sect. 16.3) with the aim to tackle the above challenges. Aiming at an infrastructure for providing an effective access to Semantic Web data, Semantic Web search engines share the following characteristics: (1) They can scale up to web-scale, that is, they are to provide an effective index for all known Semantic Web data published on the Web. Instead of directly answering queries to Semantic Web data, they use their global index to filter the relevant dataset to be used to answer queries. (2) They can provide ranking to help users deal with alternative data, and thus better assist the selection of ontologies or semantic documents of different qualities on the Web. (3) They can provide advanced "semantic-based" services to human users and computer applications, and thus enable computer-assisted search-then-query processes. In this way, they help human users better leverage the automated processing of information to conduct intelligent filtering and integration tasks.

In order to clarify the scope of Semantic Web search engines, in the following are briefly presented other categories of systems that partly share the goal of data accessibility with Semantic Web search engines.

Database systems and knowledge-based systems generally focus on answering questions using well-structured knowledge stored in closed databases or knowledge bases. The typical input is a query encoded in a formal language, such as KIF (http://www-ksl.stanford.edu/knowledge-sharing/kif/) or SQL (http://en.wikipedia.org/wiki/SQL); the typical output are variable bindings that answer the query using the stored data or knowledge. Recent advances on natural language processing (NLP) technologies such as Controlled Natural Language [10] have been used to help users in composing structured queries using natural language. Existing triple store systems can be classified as database systems or knowledge base systems depending on whether inference (e.g., RDFS or OWL inference) is executed to answer queries on data encoded in RDF graph and the corresponding ontologies. SPARQL queries are used as data access interface and the query results with bindings to RDF resources and triples are typical output. It is notable that SPARQL by itself does not encode any inference requirements, and most triple stores provide SPARQL interface with various back-end inference capability on RDFS semantics and OWL semantics. Triples store queries, database queries, and knowledge base queries share similar focus on a limited scope of data even though they could be in huge volumes, and the results are expected to be complete and sound.

Web Search and Semantic Search focus on filtering relevant text documents. Using keyword-based queries, they return documents that, in the basic case of Web search, simply contain the keywords. Semantic Search extends this conventional scenario by adding some semantic components to better exploit the intended meaning of the keywords, as well as the semantic content of the documents being searched. There have been a number of systems implementing a variety of tasks that relate to Semantic Search. For example, computer-assisted semantic query expansion based on latent semantic analysis is used to improve search results. In this way, Cuil.com presents follow-up drill down links by understanding what users want (note that on September 17, 2010 the Cuil servers were permanently taken offline). Semantic tags and annotations can also be attached to a document to better identify its content. These semantic indexes for documents can be manually entered or automatically extracted from the semantic analysis of the documents (see, e.g., PowerSet.com).

In comparison, Semantic Web search engines focus on Semantic Web data published on the open Web. They are specialized to search for documents or objects published on the Web using standard Semantic Web languages. They do not try to answer the queries directly like triple stores, but return relevant data to answer queries. Generally, they take keywords with simple constraints (e.g., restricting to particular types of entities) as input, although more formal query and exploration mechanisms are often available. Their goal is to provide a simple access point for these data, acting like classical search engines do for Web documents, but retrieving and delivering the URLs or materializations of the relevant Semantic Web data, and providing a basic Web-service infrastructure for applications to make use of these data and knowledge.

16.1.3 Abstract Specification

There are a number of initiatives that have emerged from the need for efficient, robust, and scalable Semantic Web search engines. While all these systems take different perspectives on the task of Semantic Web search, have different focuses, and are based on different assumptions, there exists a common ground that relates them to each other. This section intends to give the specification of this common base for Semantic Web search engines.

A Semantic Web search engine is a system that collects, indexes, and analyzes Semantic Web documents to provide search and querying mechanisms. Semantic Web documents are documents containing information encoded using standard Semantic Web languages such as RDF, RDFS, and OWL.

➤ *Figure 16.1* gives a general overview of the common activities of Semantic Web search engines. Not all of these components are present in all the search engines. For example, some systems rely only on manual submissions of semantic documents and do not use a crawler. However, this provides a general framework to which existing systems can be related and distinguished according to the way they implement the included components.

Crawling. Crawling is an essential task for systems with ambition to provide access to the whole set of semantic documents available on the Web. To some extent, crawling here is very similar to crawling for Web documents. However, the links that are followed by the crawler can be different (imports, explicit references through namespaces, etc.) Also, crawlers in Semantic Web search engines can exploit different sources of information to locate documents. For example, specific extensions of the sitemap mechanism have been developed (http://sw.deri.org/2007/07/sitemapextension/), as well as formats to describe

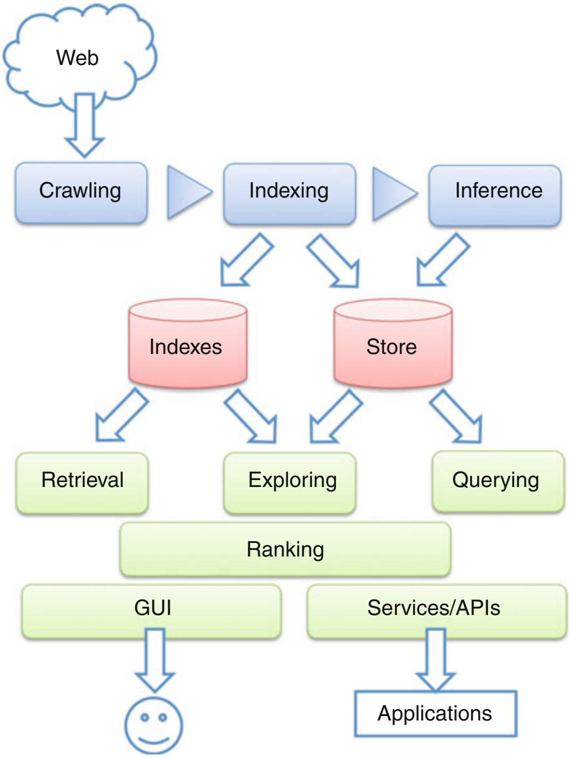


Fig. 16.1

High-level view of the activities of a Semantic Web search engine

semantic datasets online (http://semanticweb.org/wiki/VoiD). A system called *PingTheSemanticWeb.com* is dedicated to alerting Semantic Web crawlers of the appearances and updates of semantic documents online. In addition to the task of locating semantic documents, many refinements can be considered, including the necessary activity of *re-crawling* for evolving documents, of *meta-crawling* using other Web search engines, as well as the management of the overall crawling process, for example, using a pipelined approach [11]. Finally, the crawler is the part of a search engine where it is decided what should count as a semantic document, and what should be the boundaries of such a document. Indeed, Semantic Web data can be searched at different levels of granularity (see *Fig. 16.2*), ranging from the universal graph of all RDF data on the Web to a single RDF triple or even the constituent terms such as a URI. Also, some search engines may be more relaxed than others with respect to what can be included in their collection, filtering out, for example, RSS (RSS Feeds are arguably Semantic Web data because they are typically treated as XML data, as the related ontology barely use Semantic Web features.)

Indexing. One of the core elements of a Semantic Web search engine is its indexing process. Indeed, classical indexing mechanisms can be used to associate semantic documents to a set of terms, but most of the existing systems enhance such indexes for full-text

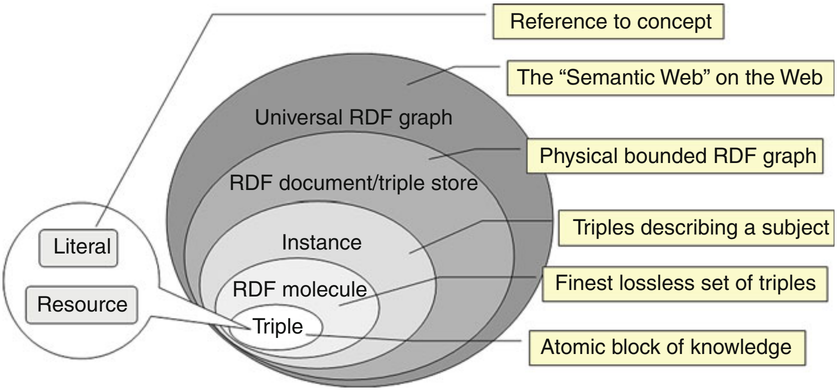


Fig. 16.2

The granularity levels range from the universal graph comprising all RDF data on the Web to individual triples and their constituent resources and literals

search with additional information such as metadata elements related to each document or indexes of the content of the documents (relations between entities) to allow for efficient querying and exploration mechanisms.

Inference. Inference can be used in a Semantic Web search engine to enhance the collected datasets and include inferred information. Heavy reasoning procedures might be used at indexing time (i.e., offline) as a one-time process, while lighter reasoning mechanisms (e.g., simple subclass transitive closure) might be realized at query time.

Ranking. As in Web search, the goal of ranking in Semantic Web search engines is to facilitate the selection of the most relevant information. However, the notion of relevance for semantic data can be more fuzzy and context dependent. Therefore, different systems adopt different approaches to the problem of ranking, from the use of simple measures originating from information retrieval [12], to more sophisticated metrics [13] and customizable ranking [14].

Retrieval. The data retrieval capabilities in different systems vary. The input ranges from keyword search to formal queries. Generally, results are URIs of Semantic Web documents, Semantic Web terms (i.e., classes and properties), and/or objects. Results can however be presented with certain amounts of additional associated metadata, and can be browsed in various ways.

Querying. While the search function is generally based on some form of keyword-based search, some systems can provide more formal ways to query the collection of documents they contain. A typical example is the use of SPARQL to allow users, but more importantly applications, to directly access the content of the documents, thus enabling their exploitation. Hence, some search engines may also play the role of global triple stores.

Navigation/Exploring. As mentioned above, Semantic Web search engines often provide browsable results, allowing the user to navigate the discovered documents (through the relations interlinking objects), to inspect the information attached to the documents or to refine the query through query expansion mechanisms. These exploration mechanisms are also very useful to applications, as they provide specific points to drill down the relevant data, for example, an agent, once it has found a class foaf:Person, can further compose a precise query on finding FOAF documents by exploring all documents that declared at least one instance of foaf:Person.

Search interface. Most of the systems provide services to agents, allowing them to directly access the metadata and search results, in addition to a graphical user interface for human users. Different technologies might be used to deliver such interfaces and the level of features provided through these services can vary from simple search mechanisms to complete APIs for the exploration and exploitation of online semantic content.

The next two sections show how the abstract specification described above is instantiated in two of the most prominent systems currently deployed.

16.1.4 Case Study 1: Swoogle

In order to support consumers to find and surf the fast-growing Semantic Web data on the Web, Swoogle [15] has been designed and implemented to complement the conventional Web search engines. › *Figure 16.3* illustrates a typical use of Swoogle in supporting web-scale Semantic Web data access. In this case, a software agent tries to answer queries using Semantic Web data on the Web via the following steps: (1) Swoogle crawls the Web for Semantic Web documents (SWDs) and Semantic Web terms (SWTs). It then builds an index for the harvested Semantic Web data and computes the corresponding rank. (2) The agent asks Swoogle's term search service using a keyword query "person" and is informed a suggested URI reference (URIref) -*foaf:Person*. (3) The agent then composes a SPARQL query using the retrieved URIrefs together with some known URIrefs. (4) The

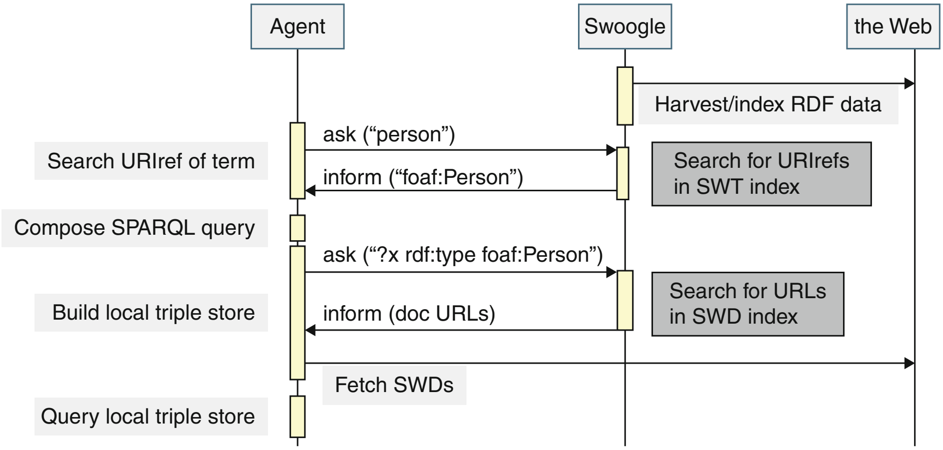


Fig. 16.3
A typical usage of Swoogle in web-scale Semantic Web data access

agent asks Swoogle's document search service for URLs of SWDs relevant to the SPARQL query. (5) The agent builds a local triple store by fetching the SWDs from the returned URLs. (6) The agent answers the SPARQL query using the integrated data in a local triple store.

16.1.4.1 Architecture

Similar to conventional Web search engines, Swoogle crawls the Web, builds indexes, computes ranks, and provides search services shown in ❯ *Fig. 16.4*. Meanwhile, Swoogle is specialized for processing Semantic Web data on the Web. In what follows, several highlighted components in this architecture are elaborated.

16.1.4.2 Crawling

In order to effectively harvest SWDs on the Web, Swoogle uses a hybrid crawler that integrates several mechanisms for discovering and harvesting Semantic Web documents on the Web. ❯ *Figure 16.5* illustrates the conceptual workflow of the hybrid crawler, and the details are explained below.

1. *Bootstrapping*. Manually submitted URLs are used to bootstrap the discovery process by providing the seeding URLs for Google-based meta-crawling and bounded HTML crawling.
2. *Google-based Meta-crawling*. Meta-crawling [16] involves directly harvesting URLs from search engines without crawling the entire Web. Google is used for several reasons: (1) It has indexed the largest number of Web documents among existing

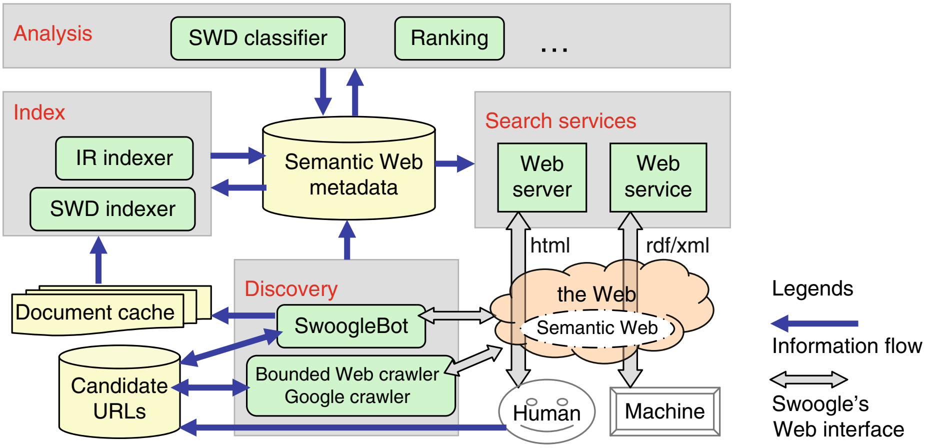


Fig. 16.4
The architecture of Swoogle

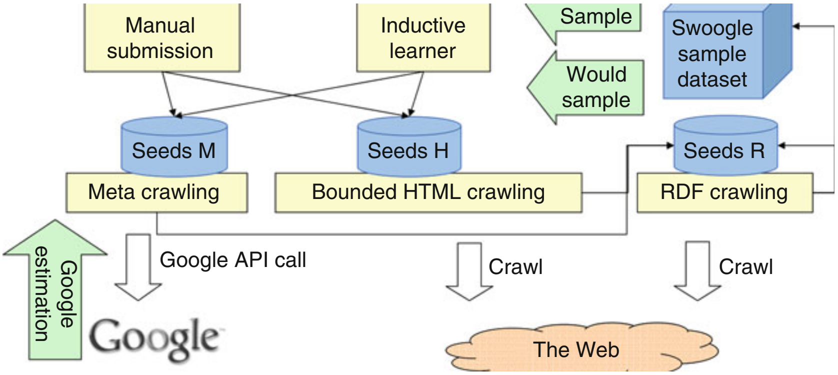


Fig. 16.5

The Swoogle system uses an adaptive Semantic Web harvesting framework with three different kinds of crawlers

Web search engines [17], (2) it does not filter Semantic Web documents out of search results, (3) it provides a Web API which is friendly to meta-crawlers, and most importantly, (4) it supports rich query constraints on both the text content and the URL of Web documents, namely "filetype," "inurl," and "site." The crawler is provided with seeds from manual bootstrapping input and enriches the seeds using the *inductive learner* that selects "good" seeds from the harvested *Swoogle sample dataset*. A "good" seed is a Google query that is believed to contribute a high percentage of SWDs, for example, most URLs returned by the query *rdf filetype:rdf* are indeed SWDs.

3. *Bounded HTML crawling. HTML crawling* (i.e., conventional Web crawling) harvests Web documents by extracting and following hyperlinks, and is useful in harvesting clusters of SWDs on the Web. The *bounded HTML crawling* imposes some thresholds (e.g., crawling depth, maximum number of URLs to visit, and minimum percentage of SWD in visited URLs) to limit search space and ensure efficiency. For example, the crawler has harvested many PML documents (SWDs) that populate instances of the Proof Markup Language (PML) by a bounded HTML crawl starting at http://iw.stanford.edu/proofs. Again, manual submission and automated inductive learner are involved in collecting seeding URLs.
4. *RDF crawling*. The *RDF crawler* enhances conventional HTML crawling by adding RDF validation and semantic hyperlink extraction components. It also visits newly discovered URLs and periodically revisits pages to keep metadata up to date. For each URL, it tries to download the content of the Web page, and then parse an RDF graph from the document using popular RDF parsers (e.g., Jena). If successful, it generates document-level metadata for the SWD and also appends the newly discovered URLS that may link to SWDs to its to-visit list.
5. *Inductive learner and Swoogle sample dataset*. The sample dataset is obtained from the metadata of the SWDs confirmed by RDF crawling. Based on the features (e.g., URL, frequency of referred Semantic Web URIs, the source website) of harvested documents

and their labels (e.g., whether they are SWD, embedded SWD or non-SWD), an automated inductive learner is used to generate new seeds for Google-based meta-crawling and bounded HTML crawling.

The crawler schedules its methods according to the following strategies: (1) Semantic Web ontologies have the highest priority since ontologies are critical for users to encode and understand Semantic Web data, (2) Semantic Web documents in RDF/XML syntax have higher priorities than Web pages that embed Semantic Web data because the former usually contain more Semantic Web data, and (3) harvesting URLs from one website is delayed where more than 10,000 SWDs have already been found at the site (e.g., liveJournal) to avoid having the catalog dominated by SWDs from a few giant websites.

16.1.4.3 Indexing

The *Indexing* component analyzes the discovered SWDs and generates the bulk of Swoogle's metadata about the Semantic Web. The metadata not only characterize the features associated with individual SWDs and SWTs, but also track the relations among them, for example, "how SWDs use/define/populate a given SWT" and "how two SWTs are associated by instantiating 'rdfs:domain' relation" [12].

The annotation metadata of a URI include the namespace and local-name extracted from the terms URI; the literal description of the term from different SWDs. The annotation metadata of SWDs include metadata about itself (such as document URL and last modified time) and its content (such as terms being defined or populated and ontology documents being imported). Moreover, Swoogle maintains relational metadata that enable users to combine keyword search and hyperlink-based surfing to locate search targets.

16.1.4.4 Ranking

Google was one of the first Web search engines to order its search results based in part on a Web page's "popularity" as computed from the Web's graph structure. This idea has turned out to be enormously useful in practice and is applicable to Semantic Web search engines. However, Google's PageRank [18] algorithm, which is based on the "random surfer model," cannot be directly used in the Semantic Web for several reasons. URIs in a document are not merely hyperlinks but semantic symbols referring to classes, instances, ontology documents, normal Web resources, etc. Semantic Web surfing is not merely random hyperlink-based surfing but rational surfing that requires understanding the semantic content of documents.

In order to rank the popularity of Semantic Web documents, the rational surfing model is adopted: a rational surfer always recursively pursues the definition of classes and properties for complete understanding of a given RDF graph. › *Figure 16.6* illustrates the rational surfing behavior of a software agent, which unfolds as follows. The agent jumps randomly to one of the accessible SWDs with uniform probability. It either terminates

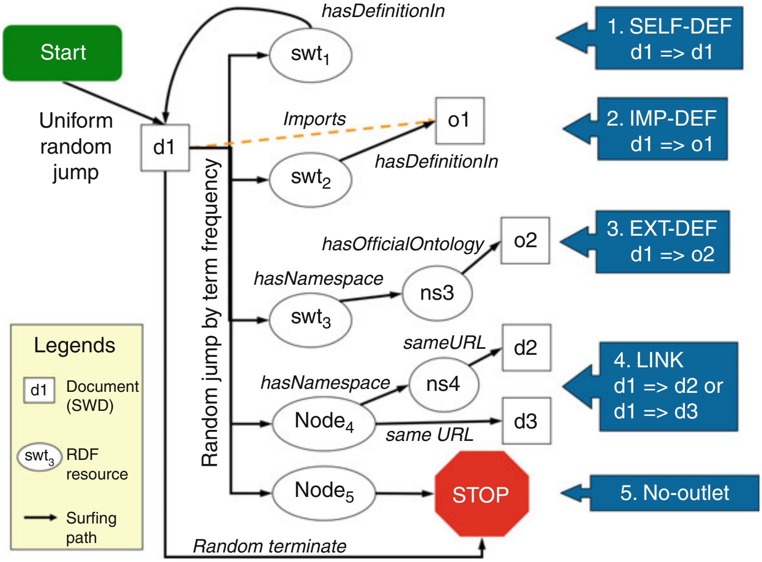


Fig. 16.6

Swoogles ranking algorithm is based on a "rational surfer model" that captures how a program might access links in processing Semantic Web documents

surfing with constant probability or chooses one RDF node in the RDF graph of the document, and the node is chosen based on its term frequency in the N-Triples version of the document. The agent either surfs to another document or terminates surfing based on the semantics of the chosen node. Paths 1 (SELF-DEF), 2 (IMP-DEF), and 3 (EXT-DEF) represent the agent pursuing a definition. If the node is not anonymous and is used as a class or property usage in the present document, the agent pursues further definition from the present document, the imported ontologies, or the ontology addressed by the namespace part of the node's URI. Path 4 (LINK) shows the hyperlink-based surfing behavior: if the node is not anonymous and is not used as a class or property, the surfer follows the URL obtained from its URI or namespace to another Semantic Web document. Path 5 (No-outlet) includes all cases when no further surfing path starts from the present node, for example, the present node is literal or anonymous, or the present node's URI links to a normal Web document.

16.1.4.5 Retrieval

The *retrieval* module provides search services to both human and software users using the indexed metadata. While queries to Web search engines return documents, the results of a Semantic Web search query can be at different levels of granularity: a Semantic Web document as well as a URI of Semantic Web terms (i.e., classes and properties). Currently, Swoogle provides two types of search services: (1) search for Semantic Web ontologies or all Semantic Web documents using keywords with additional query constraints, and

(2) search for Semantic Web terms using keywords with additional query constraints. Keywords are used to match the text parsed from the URI, labels, comments of a document, or a term. Additional query constraints can be used to filter the results using the indexed metadata, for example, only find SWTs defined as OWL class, only find SWTs defined using FOAF namespace, only find SWDs published at http://inference-web.org.

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 [12].

16.1.4.6 Archive

Like most search engines, Swoogle keeps a cache of the publicly available Semantic Web documents it indexed. Furthermore, Swoogle goes beyond this in two ways. First, it also maintains a copy of each documents representation as a set of triples, a more useful form for programs and agents. Second, and more significantly, Swoogle maintains an archive of all of the current and old versions of each Semantic Web document in its index. The resulting Semantic Web Archive (http://swoogle.umbc.edu/index.php?option=com_swoogle_service&service=archive) can be used by researchers to study how ontologies evolve, to track the growth of documents containing RDF data or to investigate the natural life cycle of the Semantic Web.

16.1.5 Case Study 2: Watson

The research on Watson originates from the observation, and anticipation, that, more and more, the way intelligent applications will be developed will change due to the availability of a large-scale, distributed body of knowledge on the Web. The dynamic exploitation of this body of knowledge introduces new possibilities and challenges requiring novel infrastructures to support the implementation of a new generation of Semantic Web applications. New mechanisms are required to enable the development of such applications, exploring large-scale semantics [19, 20]:

Finding the relevant sources: The ability to locate dynamically the sources containing relevant semantic information is a prerequisite for applications that aim to leverage the use of online knowledge. This feature is important because, in such applications, the relevance of a particular resource to a problem-solving need cannot be judged at design time.

Selecting the appropriate knowledge: From the set of previously located semantic documents, the appropriate knowledge has to be selected based on application-dependent criteria, such as the quality of the data and its adequacy to the task at hand.

Exploiting heterogeneous knowledge sources: When reusing online semantic information, no assumption can be made on the ontological nature of the elements that are manipulated. Hence the process needs to be generic enough so that it can make use of any online semantic resource. In addition, as in the case of the aforementioned tasks of finding and selecting semantic resources, this activity must also be carried out at runtime.

Combining ontologies and resources: It cannot be expected that one unique source of knowledge will provide all the required elements for a given application. Therefore, it is often necessary for next-generation Semantic Web applications to select and integrate partial fragments of knowledge from different sources, so that they can be exploited jointly.

Watson is a gateway to the Semantic Web: it collects, analyzes, and gives access to ontologies and semantic data available online. Its objective is to support the development of this new generation of Semantic Web applications that dynamically select, combine, and exploit the knowledge published on the Semantic Web.

16.1.5.1 Architecture

The role of a gateway to the Semantic Web is to provide an efficient access point to online ontologies and semantic data. Therefore, such a gateway realizes three main activities: (1) it collects the available semantic content on the Web, (2) analyzes it to extract useful metadata and indexes, and (3) implements efficient query facilities to access the data. While these three tasks are generally at the basis of any classical Web search engine, their implementation is rather different when dealing with semantic content as opposed to Web pages.

To realize these tasks, Watson is based on a number of components depicted in ➲ *Fig. 16.7*, relying on existing, standard, and open technologies. Locations of existing semantic documents are first discovered through a *crawling and tracking* component, using in particular Heritrix, the Internet Archive's Crawler (http://crawler.archive.org/). The *Validation and Analysis component* is then used to create a sophisticated system of indexes for the discovered documents, using the Apache Lucene indexing system (http://lucene.apache.org/). Based on these indexes, a core API is deployed that provides all the functionalities to search, explore, and exploit the collected semantic documents. This API also links to the *Revyu.com* Semantic Web–based reviewing system to allow users to rate and publish reviews on ontologies.

16.1.5.2 Collecting Semantic Content: Crawling the Semantic Web

The goal of the crawling task in Watson is to discover locations of semantic documents and to collect them. Classical Web crawlers can be used, but they need to be adapted to take into account the fact that the crawler is not dealing only with Web pages, but also with semantic content.

Sources: Different sources are used by the crawler of Watson to discover ontologies and semantic data (Google, Swoogle, http://pingthesemanticweb.com/, etc.). Specialized crawlers were designed for these repositories, extracting potential locations by sending

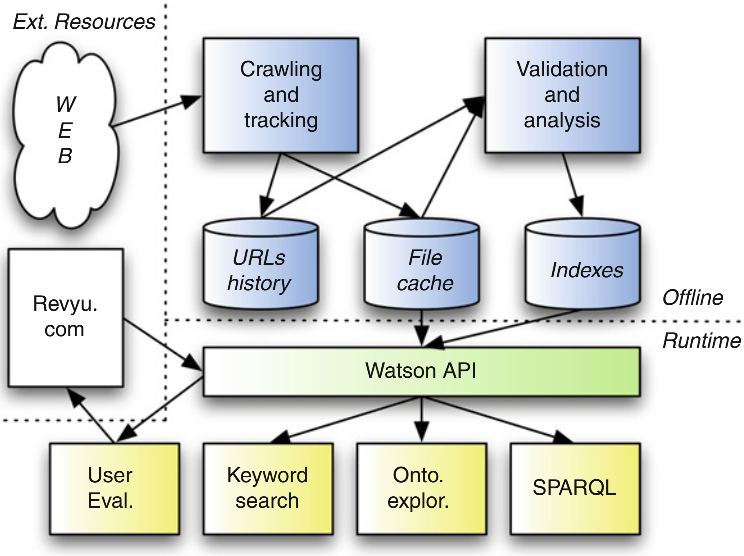


Fig. 16.7
Overview of the Watson architecture

queries that are intended to be covered by a large number of ontologies. For example, the keyword search facility provided by Swoogle is exploited with queries containing terms from the most common words in the English language. Another crawler heuristically explores Web pages to discover new repositories and to locate documents written in certain ontology languages (e.g., by including "filetype:owl" in a query to Google). Finally, already collected semantic documents are frequently re-crawled, to discover evolutions of known semantic content or new elements at the same location.

Filters: Once located and retrieved, these documents are filtered to keep only the elements that characterize the Semantic Web. In particular, to keep only the documents that contain semantic data or ontologies, the crawler eliminates any document that cannot be parsed by Jena (http://jena.sourceforge.net/). In that way, only RDF-based documents are considered. Furthermore, a restriction exists, which imposes that all RDF-based semantic documents be collected with the exception of RSS. The reason to exclude these elements is that, even if they are described in RDF, RSS feeds represent semantically weak documents, relying on RDF Schema more as a way to describe a syntax than as an ontology language.

16.1.5.3 Analyzing Semantic Content: Validation, Indexing, and Metadata Generation

Many different elements of information are extracted from the collected semantic documents: information about the entities and literals they contain, about the employed languages, about the relations with other documents, etc. This requires analyzing the content of the retrieved documents in order to extract relevant information (metadata) to be used by the search functionality of Watson.

Simple Metadata: Besides trivial information, like the labels and comments of ontologies, some of the metadata that are extracted from the collected ontologies influence the way Watson is designed. For instance, there are several ways to declare the URI of an ontology: as the namespace of the document, using the `xml:base` attribute, as the identifier of the ontology header, or even, if it is not declared, as the URL of the document. URIs are supposed to be unique identifiers in the scope of the Semantic Web. However, two ontologies that are intended to be different may declare the same URI [10, 21]. For these reasons, Watson uses internal identifiers that may differ from the URIs of the collected semantic documents. When communicating with users and applications, these identifiers are transformed into common, nonambiguous URIs.

Content: Another important step in the analysis of a semantic document is to characterize it in terms of its content. Watson extracts, exploits, and stores a large range of declared metadata or computed measures, like the employed languages/vocabularies (RDF, RDFS, OWL, DAML + OIL), information about the contained entities (classes, properties, individuals and literals), or measures concerning the richness of the knowledge contained in the document (e.g., the expressiveness of the employed language, the density of the class definitions, etc.). By combining these elements of information, Watson can decide whether or not a particular document should be treated as a semantically rich ontology. These elements are then stored and exploited to provide advanced, quality-related filtering, ranking, and analysis of the collected semantic content.

Relations between semantic documents: In the previous paragraphs, the analysis task was to extract metadata concerning one particular semantic document. In addition, a core aspect in the design of Watson concerns the exploitation of relations between semantic documents. The retrieved ontologies are inspected in order to extract information linking to other semantic documents. There are several semantic relations between ontologies that have to be followed (e.g., `owl:imports`, `rdfs:seeAlso`, `namespaces`, `derefenceable URIs`). Besides providing useful information about the considered documents, the results of this task are also used to extract potential locations of other semantic documents to be crawled.

In addition to declared semantic relations like `owl:imports`, the aim is also to compute implicit relations that can be detected by comparing ontologies. Equivalence is one of the most obvious of these relations, which is nevertheless crucial to detect. Indeed, detecting duplicated knowledge ensures that redundant information is not stored and that duplicated results are not presented to the user. On the same basis, several other relations are considered relying on particular notions of similarity between ontologies (inclusion, extension, overlap, etc.). Combined with other information from the crawler (e.g., date of discovery, of modification), these relations make possible the study and characterization of the evolution of ontologies on the Web through their different versions.

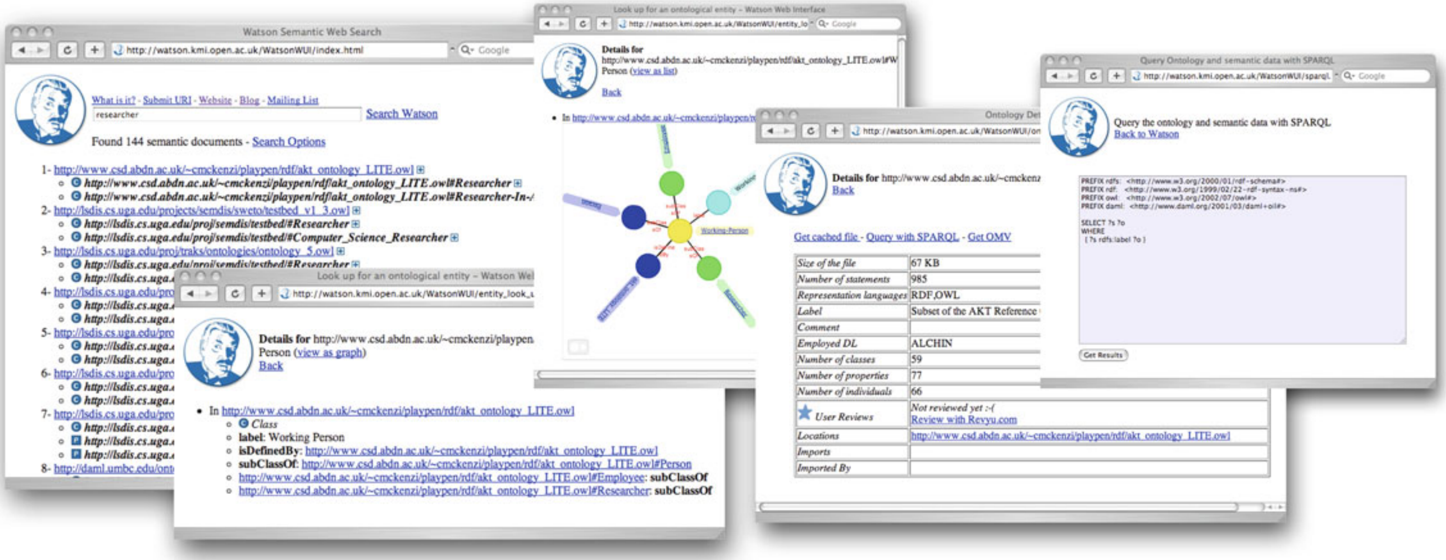
16.1.5.4 Web Interface: Search, Navigation, and Exploration

Even if the first goal of Watson is to support semantic applications, it is important to provide Web interfaces that facilitate the access to ontologies for human users. Users may have different requirements and different levels of expertise concerning semantic technologies. For this reason, Watson provides different "perspectives," from the most simple keyword search, to sophisticated queries using SPARQL (see ❯ *Fig. 16.8*). It can be accessed at the following address http://watson.kmi.open.ac.uk/.

Keyword search: The keyword search feature of Watson is similar in its use with usual Web or desktop search systems. The set of keywords entered by the user is matched to the local names, labels, comments, or literals of entities occurring in semantic documents. A list of matching ontologies is then displayed with, for each ontology, some information about it (language, size, etc.) and the list of entities matching each keyword. The search can also be restricted to consider only certain types of entities (classes, properties, individuals) or certain descriptors (labels, comments, local names, literals).

Ontology summaries: In order to facilitate the assessment and selection of ontologies by users, it is crucial to provide overviews of ontologies that are easy to read and understand, both at the level of the automatically extracted metadata about them, as well as at the level of their content. For each collected semantic document, Watson provides a page that summarizes essential information such as the size of the document (in bytes, triples, number of classes, properties, and individuals), the language used (OWL, RDF-S and DAML + OIL, as well as the underlying description logic), the links with other documents (through imports), and the reviews from users of Watson. Providing an appropriate overview of the content of an ontology or a semantic document is a difficult task. The complete graph of the content would not be really convenient for the user, and the natural language description contained in the comment about the ontology is rarely present, and generally not precise enough to help understanding the information formalized within this ontology. In other terms, there is a need to summarize ontologies, providing concise descriptions of the most important elements they contain. Peroni et al. [22] present a method to automatically extract the key concepts of an ontology using a variety of dimensions. The key concepts of an ontology are the concepts that are considered the best descriptors of the ontology by human users. In Watson, this work is used to generate small graphs, showing the six first key concepts of each ontology and an abstract representation of the existing relations between these concepts (see the example in ❯ *Fig. 16.9*). These visual summaries of ontologies provide a convenient way to obtain a quick overview of the considered ontology, which can be completed by a more precise and detailed exploration of the ontology if necessary.

Ontology exploration: One principle applied to the Watson interface is that every URI is clickable. A URI displayed in the result of the search is a link to a page giving the details of either the corresponding ontology or a particular entity. Since these descriptions also show relations to other elements, this allows the user to navigate among entities and ontologies. It is therefore possible to explore the content of ontologies, navigating through the relations between entities as well as to inspect ontologies and their metadata.



researcher

Found 144 semantic documents - Search Options

1- http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl
- *http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl#Researcher*
- *http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl#Researcher-In-*

2- http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_3.owl
- *http://lsdis.cs.uga.edu/proj/semdis/testbed/#Researcher*
- *http://lsdis.cs.uga.edu/proj/semdis/testbed/#Computer_Science_Researcher*

3- http://lsdis.cs.uga.edu/proj/traks/ontologies/ontology_5.owl
- *http://lsdis.cs.uga.edu/proj/semdis/testbed/#Researcher*

Details for http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl#W
Person (view as list)
Back

- In http://www.csd.abdn.ac.uk/~cmckenzi/playpen/

Details for http://www.csd.abdn.ac.uk/~cmckenzi/playpen
Person (view as graph)
Back

- In http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl
  - *Class*
  - **label**: Working Person
  - **isDefinedBy**: http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl
  - **subClassOf**: http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl#Person
  - http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl#Employee: **subClassOf**
  - http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl#Researcher: **subClassOf**

Details for http://www.csd.abdn.ac.uk/~cmckenz
Back

Get cached file - Query with SPARQL - Get OMV

Size of the file	67 KB
Number of statements	985
Representation languages	RDF,OWL
Label	Subset of the AKT Reference
Comment	
Employed DL	ALCHIN
Number of classes	59
Number of properties	77
Number of individuals	66
★ *User Reviews*	*Not reviewed yet :-(* Review with Revyu.com
Locations	http://www.csd.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl
Imports	
Imported By	

Query the ontology and semantic data with SPARQL
Back to Watson

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX daml:  <http://www.daml.org/2001/03/daml+oil#>

SELECT ?s ?o
WHERE
 { ?s rdfs:label ?o }
```

Fig. 16.8
Overview of the Watson Web interface

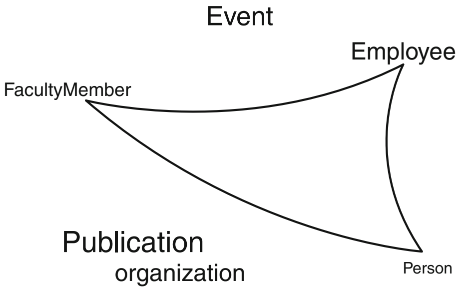


Fig. 16.9

Key concept–based visual summary of the ontology http://swrc.ontoware.org/ontology/portal

SPARQL. A SPARQL endpoint has been deployed on the Watson server and is customizable to the semantic document to be queried. A simple interface allows one to enter a SPARQL query and to execute it on the selected semantic document. This feature can be seen as the last step of a chain of selection and access tasks using the Watson Web interface. Indeed, keyword search and ontology exploration allow the user to select the appropriate semantic document to be queried. The next step is to extend this feature to be able to query not only one semantic document at a time, but also to automatically retrieve the semantic data useful for answering the query.

16.1.5.5 The Watson API

The core components of Watson are the services and API it provides to support the development of next-generation Semantic Web applications (see *Fig. 16.10*). Indeed, Watson deploys a number of Web Services and a corresponding API allowing applications to:

- Find Semantic Web documents through sophisticated keyword-based search, allowing applications to specify queries according to a number of parameters (type of entities, level of matching of the keywords, etc.)
- Retrieve metadata about these documents, for example, size, language, label, logical complexity, etc.
- Find specific entities (classes, properties, individuals) within a document
- Inspect the content of a document, that is, the semantic description of the entities it contains
- Apply SPARQL queries to Semantic Web documents

In sum, Watson's API provides a number of advantages. In Watson, it is considered that any piece of information that has been collected should be made available, so that applications are provided with as much information as possible. Also, the comprehensive set of functionalities exposed by the API allows any application to use online semantic

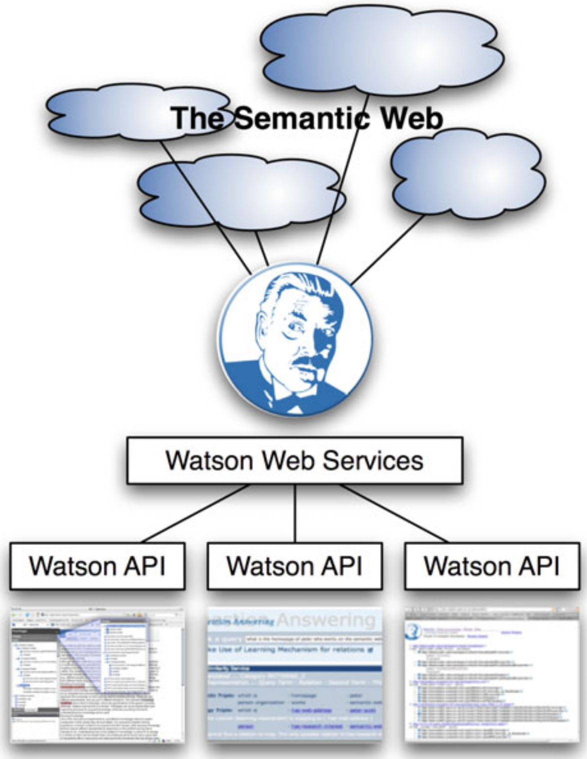


Fig. 16.10

Using the Watson API to build Semantic Web applications

data in a lightweight fashion, without even having to download the corresponding semantic documents. The content of a semantic document is processed and indexed by Watson so that it can be accessed by applications at runtime, without requiring sophisticated mechanisms and large resources.

The combination of mechanisms for searching semantic documents (keyword search), retrieving metadata about these documents and querying their content (e.g., through SPARQL) provides all the necessary elements for applications to select and exploit online semantic resources. Moreover, the Watson Web Services and API are in constant evolution to support the requirements of novel applications. In particular, an initial set of measures, which evaluate the complexity and richness of ontologies, is currently being used for ranking. A more flexible framework combining both automatic metrics for ontology evaluation and user evaluation is being developed to allow for a more customizable selection mechanism. Another important direction concerns the detection of semantic relations between ontologies to support their combination. Indeed, while a simple duplicate detection mechanism is already in place, more advanced mechanisms need to be considered to efficiently discover fine-grained relations such as extension, version, or compatibility.

16.2 Example Applications: Semantic Web Search Engines in Action

16.2.1 Semantic Web Search Engines as Development Platforms

A number of applications relying on Watson, Swoogle, and other Semantic Web search engines have been developed and provide demonstrators of the possibilities offered by exploiting the Semantic Web. This section describes a few selected applications in different categories (services, ontology and semantic data management tools, end-user applications) with the aim of providing an overview of the variety of tasks that can be achieved nowadays with the Semantic Web. More details can be found in [19, 23].

16.2.1.1 Scarlet: Relation Discovery

Scarlet (http://scarlet.open.ac.uk/) follows the paradigm of automatically selecting and exploring online ontologies to discover relations between two given concepts. For example, when relating two concepts labeled *Researcher* and *AcademicStaff,* Scarlet, using Watson, (1) identifies (at runtime) online ontologies that can provide information about how these two concepts interrelate and then (2) combines this information to infer their relation. Two increasingly sophisticated strategies were investigated to discover and exploit online ontologies for relation discovery. The first strategy, S1, derives a relation between two concepts if this relation is defined within a single online ontology, for example, stating that $Researcher \sqsubseteq AcademicStaff$. The second strategy, S2, addresses those cases in which no single online ontology states the relation between the two concepts, by combining relevant information which is spread over two or more ontologies, for example, that $Researcher \sqsubseteq ResearchStaff$ in one ontology and that $ResearchStaff \sqsubseteq AcademicStaff$ in another. To support this functionality, Scarlet relies on Watson to access online ontologies.

Scarlet originates from the design of an ontology matcher that exploits the Semantic Web as a source of background knowledge to discover semantic relations (mappings) between the elements of two ontologies. This matcher was evaluated in the context of aligning two large, real-life thesauri: the UNs AGROVOC thesaurus (40 K terms) and the United States National Agricultural Library thesaurus NALT (65 K terms) [24]. The matching process performed with both strategies resulted in several thousand mappings, using several hundred online ontologies, with an average precision of 70%.

16.2.1.2 Swoogle Ontology Dictionary

Swoogle Ontology Dictionary is an 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:

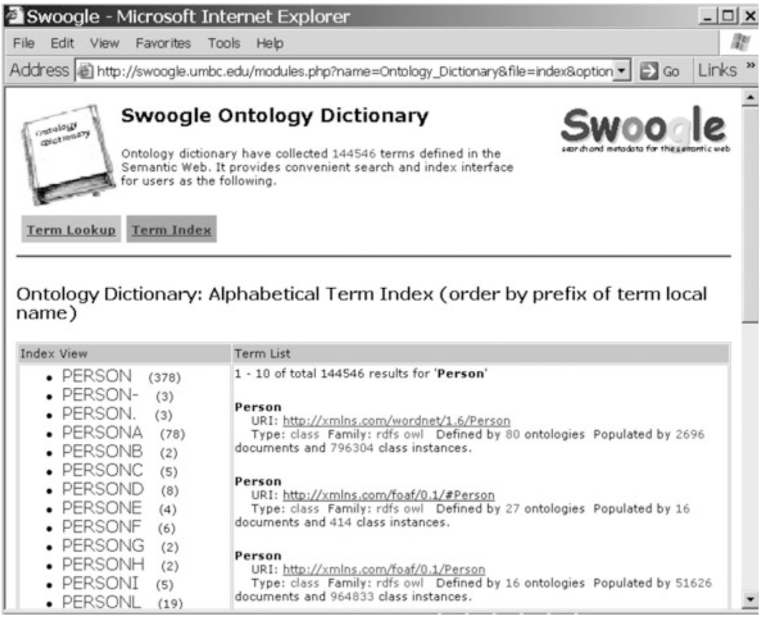
- It builds a comprehensive view of the Semantic Web vocabulary and breaks the (unnecessary) physical boundary imposed by Semantic Web ontologies. There are two well-known drawbacks of using ontology documents to group Semantic Web terms: (1) Semantic Web terms defined in one Semantic Web ontology may be instantiated in quite different frequencies, for example, *owl:versionInfo* is far less instantiated than *owl:Class* in the Semantic Web; and (2) 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 collects 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.

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

- *Term Search* is essentially a web interface based on the 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 ❯ *Fig. 16.11*, 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 (using case-insensitive string matching here). In the matched-term-list view, all terms matching the current prefix are listed.

16.2.1.3 Sig.Ma

Sig.Ma [25] (http://sig.ma) is a service built on top of the Sindice [26] Semantic Web search engine. Sindice indexes very large quantities of information from the Web, especially coming from the linked data community. Sig.Ma relies on Sindice to provide an aggregated view on the available semantic data for a given entity or resource. Starting from a simple keyword query supposed to describe the entity to look up; Sig.Ma displays the properties of the corresponding entities present in a large variety of linked data sources, as well as the correspondences between each piece of data and the sources where it originated. For example, using the name of a person as a starting point, Sig.Ma can show the location, photos, workplace, contact details, and birthday of this person, each piece of information potentially coming from a different source. Of course, noise could easily



Swoogle Ontology Dictionary

Ontology dictionary have collected 144546 terms defined in the Semantic Web. It provides convenient search and index interface for users as the following.

Ontology Dictionary: Alphabetical Term Index (order by prefix of term local name)

Index View

- PERSON (378)
- PERSON- (3)
- PERSON. (3)
- PERSONA (78)
- PERSONB (2)
- PERSONC (5)
- PERSOND (8)
- PERSONE (4)
- PERSONF (6)
- PERSONG (2)
- PERSONH (2)
- PERSONI (5)
- PERSONL (19)

Term List

1 - 10 of total 144546 results for '**Person**'

Person
URI: http://xmlns.com/wordnet/1.6/Person
Type: class Family: rdfs owl Defined by 80 ontologies Populated by 2696 documents and 796304 class instances.

Person
URI: http://xmlns.com/foaf/0.1/#Person
Type: class Family: rdfs owl Defined by 27 ontologies Populated by 16 documents and 414 class instances.

Person
URI: http://xmlns.com/foaf/0.1/Person
Type: class Family: rdfs owl Defined by 16 ontologies Populated by 51626 documents and 964833 class instances.

Fig. 16.11

The alphabetical term index interface

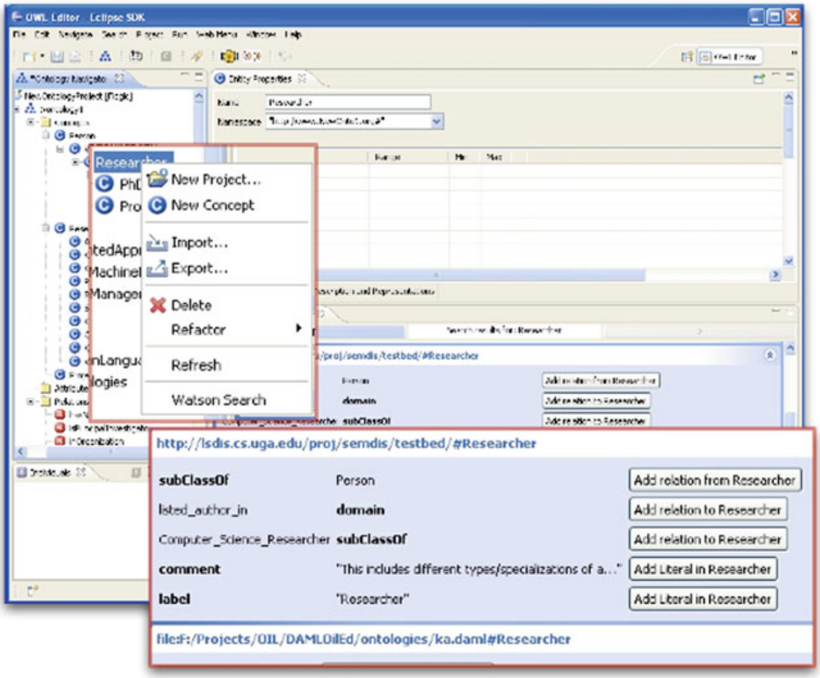
appear in the results, due to the potential ambiguity of the initial query. Sig.Ma allows the user to customize the view by refining the list of sources, removing the ones which do not match the initial intent of the query.

A point worth noticing is that Sig.Ma is not only an application itself, but also provides a base for other applications. Each view, even customized, is associated with a Web address (a URI). An API and a widget are also available that give access to the functionalities of Sig.Ma to other applications.

16.2.1.4 The Watson Plug-In for Knowledge Reuse

Ontology reuse is a complex process involving activities such as searching for relevant ontologies for reuse, assessing the quality of the knowledge to reuse, selecting parts of it and, finally, integrating it in the current ontology project. As the Semantic Web provides more and more ontologies to reuse, there is an increasing need for tools supporting these activities.

The Watson plug-in (http://watson.kmi.open.ac.uk/editor_plugins.html) (see *Fig. 16.12*) aims to facilitate knowledge reuse by integrating the search capabilities of Watson within the environment of an ontology editor (the NeOn Toolkit, http://neon-toolkit.org). The resulting infrastructure allows the user to perform all the steps necessary for the large-scale reuse of online knowledge within the same environment where this knowledge is processed and engineered.



http://lsdis.cs.uga.edu/proj/semdis/testbed/#Researcher

subClassOf	Person	Add relation from Researcher
listed_author_in	**domain**	Add relation to Researcher
Computer_Science_Researcher	**subClassOf**	Add relation to Researcher
comment	"This includes different types/specializations of a..."	Add Literal in Researcher
label	"Researcher"	Add Literal in Researcher

file:F:/Projects/OIL/DAMLOilEd/ontologies/ka.daml#Researcher

Fig. 16.12
The Watson Plug-in for ontology editors

In practice, the Watson plug-in allows the ontology developer to find, in existing online ontologies, descriptions of the entities present in the currently edited ontology (i.e., the *base* ontology), to inspect these descriptions (the statements attached to the entities) and to integrate these statements into the base ontology. For example, when extending the base ontology with statements about the class *Researcher*, the Watson plug-in identifies, through Watson, existing ontologies that contain relevant statements such as:

- Researcher is a subclass of *AcademicStaff*
- *PhDStudent* is a subclass of Researcher
- Researcher is the domain of the property *isAuthorOf*

These statements can be used to extend the edited ontology, integrating them to ensure, for example, that the class Researcher becomes a subclass of a newly integrated class *AcademicStaff*.

16.2.1.5 Swoogle-Based Triple Shop

Triple Shop [27] was developed to better assist users to utilize the search results of Swoogle. It worked as follows: Swoogle would present query results (URIs) to the user, and then the user could check URIs to be added to his or her shopping cart. Eventually, a user could check out, have all URIs loaded into a triple store and be presented with an interface for issuing SPARQL queries. This utility proved to be an extremely useful tool in integrating scientific data. Below are some key features:

- Finding datasets. A dataset finder is a service that implements the Swoogle-assisted data access process by facilitating the completion of an incomplete SPARQL-ish query. Besides manually specifying the URIs of RDF resources, users can simply use English terms to refer to RDF resources in the WHERE clause of a SPARQL query. This service will search Swoogle for appropriate URIs to substitute the English terms in the query, and the user can then select one from the alternative resulting URIs. Users can also leave dataset specification empty, that is, without specifying the FROM clause. Again, the service will search Swoogle to suggest relevant SWDs to answer the query. It is notable that the search for SWDs and SWTs can be refined in a number of ways. Constraints can be placed on the domain of a URI, and on the namespaces that it uses.
- Inference. After constructing a dataset, the user can specify a level of reasoning to be performed in executing the query. Choices range from no reasoning, through RDFS, to OWL.
- Dataset persistence and reuse. A user can save a dataset on the Triple Shop server, tag a dataset, search for existing tagged datasets, and add tags to existing datasets. Each dataset can be stored as a list of URLs of SWDs, or be materialized into a merged RDF graph in triple store.

Triple Shop has been used in ELVIS (the Ecosystem Location Visualization and Information System), which is a suite of tools for constructing food webs for a given location. ELVIS is motivated by the belief that food web structure plays a role in the success or failure of potential species invasions. Because very few ecosystems have been the subject of empirical food web studies, response teams are typically unable to get quick answers to questions like what are likely prey and predator species of the invader in the new environment? The ELVIS tools seek to fill this gap. ELVIS functionality is exposed as a collection of Web Services, and all input and output data are expressed in OWL, thereby enabling its integration with other Semantic Web resources.

Bioinformatic data in ELVIS are encoded in RDF and cover the following categories: (1) species distribution data compiled by the California node of the National Biological Information Infrastructure; (2) trophic data compiled from over 250 datasets; (3) the complete contents of Animal Diversity Web (ADW), a popular online encyclopedia [48]; and (4) a collection of lists designating species as being invasive in particular regions.

With the available datasets in ELVIS, researchers can verify their hypotheses on the complex relations in a food web. The complex relations can be mapped to a query on the RDF data in ELVIS. As the researchers may not necessarily know URIs for all the terms or know which datasets are relevant, Triple Shop can assist completing a query with the help of Swoogle, gathers/integrates all triples that might be relevant to the query, and will do forward-chaining inference to generate all implied triples when appropriate. This process may take anywhere from seconds to hours. When it is complete, the researchers can see query results, and share the resulting integrated dataset with colleagues in a persistent manner.

16.2.1.6 Evolva: Ontology Evolution Using Background Knowledge

Ontologies form the backbone of Semantic Web–enabled information systems. Today's organizations generate huge amounts of information daily, thus ontologies need to be kept up to date in order to reflect the changes that affect the life cycle of such systems (e.g., changes in the underlying datasets, a need for new functionalities, etc.). This task, described as the "timely adaptation of an ontology to the arisen changes and the consistent management of these changes," is called *ontology evolution* [28]. While it seems necessary to apply such a process consistently for most of the ontology-based systems, it is often a time-consuming and knowledge-intensive task, as it requires a knowledge engineer to identify the need for change, perform appropriate changes on the base ontology, and manage its various versions.

Evolva (an overview of Evolva can be found in [29, 30]) is an ontology evolution system starting from external data sources (text documents, folksonomies, databases, etc.) that form the most common means of storing data. First, a set of terms are extracted from these sources as potentially relevant concepts/instances to add to the ontology, using common information extraction methods. Evolva then makes use of Watson (through the intermediary of Scarlet) to find external sources of background knowledge to establish relations between these terms and the knowledge already present in the ontology, providing in this way the means to integrate these new terms in the ontology. For this purpose, a relation discovery process was devised, that combines various background knowledge sources with the goal of optimizing time-performance and precision.

16.2.1.7 Wahoo/Gowgle: Query Expansion

Wahoo and Gowgle (http://watson.kmi.open.ac.uk/wahoo and http://watson.kmi.open.ac.uk/gowgle) are two demonstrators, showing how Watson can be used for a simple application to perform query expansion in a classical Web search engine. For example, when given the keyword *developer*, such a tool could find out that in an ontology, there is a subclass *programmer* of *developer* and could therefore suggest this term as a way to specify the query to the Web search engine. Without Watson, this would require one to integrate one or several ontologies about the domain of the queries and an infrastructure to store them, explore them, and query them. However, if the considered search engine is a general Web search engine, such as Google or Yahoo!, the domain of the queries cannot be predicted: the appropriate ontology can only be selected at runtime, depending on the query that is given. In addition, this application would require a heavy infrastructure to be able to handle large ontologies and to query them efficiently. Gowgle and Wahoo rely on Semantic Web ontologies explored using Watson instead.

The overall architecture of these applications is made of a Javascript/HTML page for entering the query and displaying the results, which communicates using the principles of AJAX with the Watson server. In the case of Gowgle, Google is used as the Web search engine and the Watson SOAP Web Services are employed for ontology exploration (http://watson.kmi.open.ac.uk/WS_and_API.html). In the case of Wahoo, Yahoo!, and the Watson REST API (http://watson.kmi.open.ac.uk/REST_API.html) are used.

Both applications use Watson to exploit online ontologies in order to suggest terms related to the query, that is, if the query contains the word *developer* (1) to find ontologies somewhere talking about the concept of developer, (2) to find in these ontologies which entities correspond to *developer*, and (3) to inspect the relations of these entities to find related terms.

16.2.1.8 SWAML

SWAML (http://swaml.berlios.de/), the Semantic Web Archive of Mailing Lists Project [31], is building a series of tools to enable the semantic publication and browsing of e-mail collections. It is able to extract e-mails from a mailbox and create a representation of these e-mails using mainly the SIOC ontology (http://sioc-project.org/). However, the information contained in the mailbox alone is not enough to organize its content. People information, for example, is present in many different sources on the Semantic Web, based on the FOAF vocabulary (http://www.foaf-project.org/). SWAML, therefore, uses Sindice to collect semantic data related to people, based on their e-mail addresses. One of the advantages of Sindice in this case is its ability to draw inferences on inverse functional properties (IFPs). Indeed, in FOAF, the relation connecting a person to his or her e-mail address is declared as an IFP, meaning that an e-mail address is associated to only one person. Therefore, whenever several resources appear to be connected to the same e-mail address, Sindice can infer that these resources refer to the same person.

16.2.1.9 PowerAqua: Question Answering

To some extent, PowerAqua (http://poweraqua.open.ac.uk/) can be seen as a straightforward human interface to any semantic document indexed by Watson. Using PowerAqua, a user can simply ask a question, like "Who are the members of the rock band Nirvana?" and obtain an answer, in this case in the form of a list of musicians (Kurt Cobain, Dave Grohl, Krist Novoselic, and other former members of the group). The main strength of PowerAqua resides in the fact that this answer is derived dynamically from the relevant datasets available on the Semantic Web.

Without going into too many details, PowerAqua first uses a Gate-based [32] linguistic component to transform a question into a set of possible "query triples," such as <person/organization, members, rock band Nirvana>. The next step consists then in locating, thanks to Watson, online semantic documents describing entities that correspond to the terms of the query triples, locating, for example, an individual called *Nirvana* in a dataset about music. During this step, WordNet (http://wordnet.pinceton.edu) is used to augment the terms in the query triples with possible synonyms. Once a collection, usually rather large, of potential candidate ontologies is found, PowerAqua then employs a variety of heuristics and a powerful matching algorithm, PowerMap [33], to try and find answers from the collection of candidate ontologies. In the example, the query triple shown above can be successfully matched to the schema <Nirvana, has_members, ?x:Musician>, which has been found in a music ontology on the Semantic Web. In more complex examples, an answer may require integrating a number of statements. For instance, to answer a query such as "Which Russian rivers flow to the Black Sea," PowerAqua may need to find information about Russian rivers, information about rivers which flow to the Black Sea and then combine the two. In general, several sources of information, coming from various places on the Web, may provide overlapping or complementary answers. These are therefore ranked and merged according to PowerAqua's confidence in their contribution to the final answer.

16.2.1.10 PowerMagpie: Semantic Browsing

PowerMagpie (http://powermagpie.open.ac.uk) is a Semantic Web browser that makes use of openly available semantic data through Watson to support the interpretation process of the content of arbitrary Web pages. Unlike its predecessor, Magpie, which relied on a single ontology selected at design time, PowerMagpie automatically, that is, at runtime, identifies and uses relevant knowledge provided by multiple online ontologies. From a user perspective, PowerMagpie is an extension of a classical Web browser and takes the form of a vertical widget displayed on top of the currently browsed Web page. This widget provides several functionalities that allow the exploration of the semantic information relevant to the current Web page. In particular, it summarizes conceptual entities relevant to the Web page. Each of the entities can then be shown in the text, where the user may initialize different ways of exploring the information space around a particular entity. In addition, the semantic information discovered by PowerMagpie, which relates the text to online semantic resources, is "injected" into the Web page as embedded annotations in RDFa. These annotations can then be stored into a local knowledge base and act as an intermediary for the interaction of different semantic-based systems.

16.2.1.11 FLOR: Folksonomy Ontology Enrichment

Folksonomies, social tagging systems such as Flickr and Delicious, are at the forefront of the Web2.0 phenomenon as they allow users to tag, organize, and share a variety of information artifacts. The lightweight structures that emerge from these tag spaces only weakly support content retrieval and integration applications since they are agnostic to the explicit semantics underlying the tags and the relations among them. For example, a search for *mammal* ignores all resources that are not tagged with this exact word, even if they are tagged with specific mammal names such as *lion*, *cow*, and *cat*. The objective of FLOR [34] is to attach formal semantics to tags, derived from online ontologies and make the relations between tags explicit (e.g., that *mammal* is a superclass of *lion*). The enrichment algorithm that has been experimentally investigated builds on Watson: given a set of tags, the prototype identifies the ontological entities (classes, properties, and individuals) that define the tags in their respective contexts. Additionally, it aims to identify formal relations between the tags (subsumption, disjointness, and generic relations) utilizing Scarlet.

The experiments [21] have led to further insights into the nature of ontologies on the Semantic Web, from which two key ones are highlighted here. First, it was found that online ontologies have a poor coverage of a variety of tag types denoting novel scientific terminology, multilingual terms, and domain-specific jargon. Secondly, it was observed that online ontologies can reflect different views and when used in combination can lead to inconsistencies in the derived structures.

16.2.1.12 The Watson Synonym Service

The Watson Synonym Service (http://watson.kmi.open.ac.uk/API/term/synonyms) is a simple service that creates a base of term clusters, where the terms of a cluster are supposed to be associated to the same sense. It makes use of the information collected by Watson in the form of ontologies to derive these clusters.

The basic algorithm to create term clusters is quite straightforward. Entities in Semantic Web ontologies all possess one and only one identifier (in a given namespace, e.g., *Person* is considered to be the identifier of http://www.example.org/onto#Person). They can also be associated to one or several labels, through the `rdf:label` property. Hence, the algorithm simply assumes that a term t_1 is a synonym of another term t_2 if t_1 and t_2 are used either as label or identifier of the same entity. The role of the synonym discovery offline algorithm is then simply to iterate through all the entities in Watson's ontologies to create clusters of terms that are used together in the identifiers or labels of entities.

Of course, the quality of the results obtained with this method is not as good as the one obtained with the complex and costly approaches that are employed to build systems such as WordNet (http://wordnet.pinceton.edu). However, the advantage of this algorithm is that its quality improves together with the growth of the Semantic Web, without requiring any additional effort for collecting the data. A high number of good synonyms are found, like in the cluster {*ending*, *death*, *termination*, *destruction*}. In addition, this method does not only find synonyms in one language, but can provide the equivalent terms in various languages, providing that multilingual ontologies exist and cover these terms. It could be argued that these are not actually synonyms (but translations) and one of the possible extensions for this tool is to make use of the language information in the ontologies to distinguish these cases.

16.2.2 Semantic Web Search Engines as Research Platforms

Semantic Web search engines are tools and infrastructure components that automatically collect, analyze, and index ontologies and semantic data available online. Besides enabling the exploitation of the Semantic Web, they can be seen as a research platform supporting the exploration of the Semantic Web to better understand its characteristics. Indeed, most of the existing systems provide statistics for the documents and Semantic Web entities they have collected (see, e.g., the statistics page of the Falcons system, http://iws.seu.edu.cn/services/falcons/statistics.jsp), but beyond basic statistics, researchers involved in the development of Semantic Web search engines were able to realize global studies of the Semantic Web landscape, using the large collections of ontologies and semantic data available through these systems.

16.2.2.1 Swoogle-Based Semantic Web Statistics

Based on the Semantic Web dataset collected by Swoogle, measures of some statistical properties of Semantic Web data were presented in [35]. This paper should be considered for precise results available at the time of its publication; however, the focus here is on demonstrating how Swoogle can be used to compute these measures, as the actual values would need to be updated to reflect the current status of the Semantic Web.

One interesting question is the size of the Semantic Web on the Web. However, this number is hard to obtain because (1) Semantic Web documents are sparsely distributed on the Web and (2) validating whether a Web document is a Semantic Web document requires nontrivial computation. Brute-force sampling, that is, measuring the size of the Web (e.g., testing 80 ports for a huge list of IP addresses) [36], is not suitable due to their unacceptable low efficiency. Analysis on the overlap of meta-search results of conventional Web search engines [17, 37] is suitable mainly because SWDs are less favored by these engines, and some even provide limited support on searching SWDs. For example, even though both support filetype search, only Google search but not MSN search supports searching for the filetype "rdf" and "owl." A Google-based meta-search is adopted for estimating SWDs based on the observation that 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.

Another interesting measure is 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. It was also noticed that the bias introduced by the dynamic SWDs could block the diversity of the Semantic Web and should be controlled. A good number of metrics have been proposed for measuring the statistical distribution of SWDs and SWTs. SWDs are the atomic containers for transferring Semantic Web data and the interfaces between the Web and the RDF graph world.

- Source of SWD. In order to measure how Semantic Web data are distributed on the Web, SWDs are grouped by their source websites. SWDs can further be grouped by the top-level domain extracted from the URLs of the website hosting the SWDs.
- 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.
- Age of SWD. SWDs could be uploaded, modified, and removed on the Web. The age of an SWD is measured by the last modified time (attached in the header of HTTP response) of its latest version.
- Size change of SWD. In order to track the size change of SWDs, snapshots of each SWD are maintained once a new version has been detected.
- Definition quality of SWD. In order to evaluate the portion of the definition in an SWD, the *ontology ratio* (OntoRatio) is calculated 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.

SWTs are also evaluated using collected data.

- Overall Meta-Usage of SWT. Analyzes the usage of SWTs in SWDs based on the combination of the six types of meta-usage identified by the WOB ontology, namely, *hasClassDefinitionIn*, *hasPropertyDefinitionIn*, *hasClassInstanceIn*, *hasPropertyInsanceIn*, *hasClassReferenceIn*, and *hasPropertyReferenceIn*.
- Definition quality of SWT. The definition of an SWT depends on its residential RDF graph that is serialized by an SWD. Again, the number of definitional triples of the SWT is counted to estimate the quality of its definition within an SWD. Usually, important classes and properties have more definitional triples.
- A common question posed by Semantic Web knowledge consumers is what kind of Semantic Web data are available. The answer to this question is given by measuring the instance space of the Semantic Web, that is, how SWTs are populated in SWDs as classes and properties, for example, the number of SWTs being populated as class (or property) by at least *m* instances,

The navigational paths in the Semantic Web are still in small amount and not enough for effective Semantic Web surfing. In the category, Navigation Quality using statistics of several important types of paths was investigated: (1) paths based on explicit import semantics, (2) paths based on inexplicit namespace reference, and (3) paths based on Link Indicators, such as the value of rdfs:seeAlso in FOAF.

16.2.2.2 Characterizing Knowledge on the Web with Watson

To give an account of the way semantic technologies are used to publish knowledge on the Web, of the characteristics of the published knowledge, and of the networked aspects of the Semantic Web, an analysis of a sample of 25,500 semantic documents collected by Watson was realized (see [38] for the details).

This analysis looked in particular into the use of Semantic Web languages and of their primitives. Watson implements a simple, but restrictive language detection mechanism. It is restrictive in the sense that it considers a document to employ a particular language only if this document actually *instantiates* an entity of the language vocabulary (any kind of description for RDF, a class for RDF-S, and a class or a property for OWL and DAML + OIL). A simple conclusion that can be drawn from this analysis is that, while the majority of the considered documents are exclusively considering factual data in RDF, amongst the ontology representation languages (RDF-S, OWL and DAML + OIL), OWL has clearly been adopted in majority.

The initial version of OWL was divided into three sub-languages, OWL Lite, OWL DL, and OWL Full, that represent different (increasing) levels of complexity (in the current version, OWL 2, these sub-languages have been replaced by *profiles*, see http://www.w3.org/TR/owl2-profiles/ see also ❯ KR and Reasoning on the Semantic Web: OWL). Another way to measure the expressivity (and so the complexity) of the language used is to consider the underlying description logic. Description logics are named according to the primitives they contain. For example, the DL of OWL Lite is $\mathcal{ALCR}_{+}\,\mathcal{HIF}\,(D)$, meaning, for example, that it allows the description of inverse relations ($\mathcal{I}$) and of limited cardinality restrictions ($\mathcal{F}$). One noticeable fact that can be derived from analyzing both the OWL Species and the description logic used in ontologies is that, while a large majority of the ontologies in the set were in OWL Full (the most complex variant of OWL, which is undecidable), most of them were in reality very simple, only using a small subset of the primitives offered by the language (95% of the ontologies where based on the $\mathcal{ALH}(\mathcal{D})$ description logic). This is consistent with conclusions obtained in [39].

Looking at the size and structure of Semantic Web documents also highlighted that a large majority of them were very simple. Indeed, a simple measure of density for RDF entities is used (measuring relations they share with other entities) and discovered that the employed collection of online semantic documents was made of a very large number of very small and very shallow structures, and of a very small number of very large and complex ontologies.

Another interesting element to consider is the duplication or URIs. Indeed, in theory, if two semantic documents are identified by the same URI, they are supposed to contribute to the same ontology, that is, the entities declared in these documents are intended to belong to the same conceptual model. However, even if this situation appears rarely (only 60 URIs of documents are “nonunique” in the considered set), in most cases, semantic documents that are identified by the same URI are not intended to be considered together. Different situations can be distinguished that lead to this problem:

Default URI of the ontology editor: http://a.com/ontology is the URI of 20 documents that do not seem to have any relation with each other, and that are certainly not meant to be considered together in the same ontology. The reason for this URI to be so popular is that it was the default namespace attributed to ontologies edited using the Protégé editor (http://protege.stanford.edu/) at the time. This problem has been reduced now by the fact that Protégé forces its users to change the URI of their ontologies.

Mistaken use of well-known namespaces: The second most commonly shared URI in the Watson repository is http://www.w3.org/2002/07/owl, which is the URI of the OWL schema. The namespaces of RDF, RDF Schema, and of other well-known vocabularies are also often duplicated. Using these namespaces as URIs for ontologies is (in most cases) a mistake that could be avoided by checking, prior to giving an identifier to an ontology, if this identifier has already been used in another ontology.

Different versions of the same ontology: A third common reason for which different semantic documents share the same URI is in situations where an ontology evolves to a new version, keeping the same URI (e.g., http://lsdis.cs.uga.edu/proj/semdis/testbed/). As it is the same ontology, it seems natural to keep the same URI, but in practice, this can cause problems in these cases where different versions coexist and are used at the same time. This leads to a need for recommendations of good practices on the identification of ontologies, that would take into account the evolution of the ontologies, while keeping different versions clearly separated.

Related to this last point, an initial experiment [40] recently investigated the use of information encoded in the URIs of the ontologies to encode versioning data, which can be extracted to trace the different versions of ontologies. It appears that many different, more or less popular conventions are used to encode such version data, from the use of version numbers (e.g., `v1.2`, `rev=3.6`) to the use of time-stamps and dates (using two or three numbers, in big endian or little endian orders). Through recognizing these patterns in URIs, many "chains" of ontology versions can be detected with varying levels of accuracy, providing an insight on how ontologies evolve on the Web.

Watson provides an efficient platform, allowing researchers to obtain an overview of the Semantic Web, to apprehend its content and development, and to analyze the way knowledge is published online. Many other elements have been, and could be analyzed concerning the Semantic Web, including the (explicit and implicit) relationships existing between documents, the coverage in terms of domains and topics, etc. [41]. The next section briefly summarizes recent work on using Watson to measure agreements and disagreements in ontologies.

16.2.2.3 Measuring Ontology Agreement and Disagreement in Watson

Ontologies are knowledge artifacts representing particular models of some particular domains. They are built within the communities that rely on them, meaning that they represent consensual representations inside these communities. However, when considering the set of ontologies distributed on the Web, many different ontologies can cover the same domain, while being built by and for different communities. Knowing which ontologies agree or disagree with others or how much a particular statement is generally agreed with in online ontologies can be very useful in many scenarios.

One way to detect whether there is a disagreement between two ontologies is to rely on the presence of logical contradictions. The two ontologies can be merged, based on mappings between their entities, and the resulting model be checked for inconsistencies and incoherences. While this approach would certainly detect some forms of disagreement, it only checks whether the ontologies disagree or not. It does not provide any granular notion of disagreement and, if no contradictions are detected, it does not necessarily mean that the ontologies agree. Indeed, while two ontologies about two completely different, nonoverlapping domains would certainly not disagree, they do not agree either. More importantly, logical contradictions are not the only way for two ontologies to disagree. Indeed, there could also be conceptual mismatches, like in the case where one ontology declares that "Lion is a subclass of Species" and the other one indicates that "Lion is an instance of Species." Even at content level, logical contradictions would not detect some form of disagreements. Indeed, the two statements "Human is a subclass of Animal" and "Animal is a subclass of Human" do not generate any incoherence. However, they disagree in the sense that, if put together, they generate results that were not expected from any of the two ontologies.

For these reasons, [42] defines two basic measures for assessing agreement and disagreement of an ontology O with a statement $s = < subject, relation, object >$:

$$\begin{aligned} agreement\ (O, s) &\rightarrow [0..1] \\ disagreement\ (O, s) &\rightarrow [0..1] \end{aligned}$$

Two distinct measures are used for agreement and disagreement so that an ontology can, at the same time and to certain extents, agree and disagree with a statement. These two measures have to be interpreted together to indicate the particular belief expressed by the ontology O regarding the statement s. For example, if $agreement(O, s) = 1$ and $disagreement(O, s) = 0$, it means that O fully agrees with s and conversely if $agreement(O, s) = 0$ and $disagreement(O, s) = 1$, it fully disagrees with s. Now, agreement and disagreement can vary between 0 and 1, meaning that O can only partially agree or disagree with s and sometimes both, when $agreement(O, s) > 0$ and $disagreement(O, s) > 0$. Finally, another case is when $agreement(O, s) = 0$ and $disagreement(O, s) = 0$. This basically means that O neither agrees nor disagrees with s, for the reason that it does not express any belief regarding the relation encoded by s.

The actual values returned for both measures, when different from 0 and 1, are not very important. They correspond to different levels of disagreement/agreement and only an order between predefined levels is needed to interpret them. The values used and the ways to compute them are given in [42].

Considering that ontologies are made of statements, extending the measures above to compute agreement and disagreement between two ontologies is relatively straightforward, using the mean of each measure for each statement of an ontology against the other ontology, in both directions and making this a normalized measure. However, while relatively simple, the two measures of agreement and disagreement between ontologies provide an interesting way to obtain an overview of a set of ontologies. Indeed, an experiment looked at the 21 ontologies returned by Watson when querying for semantic documents containing a class with the term *SeaFood* in its ID or label, and computed the agreement and disagreement

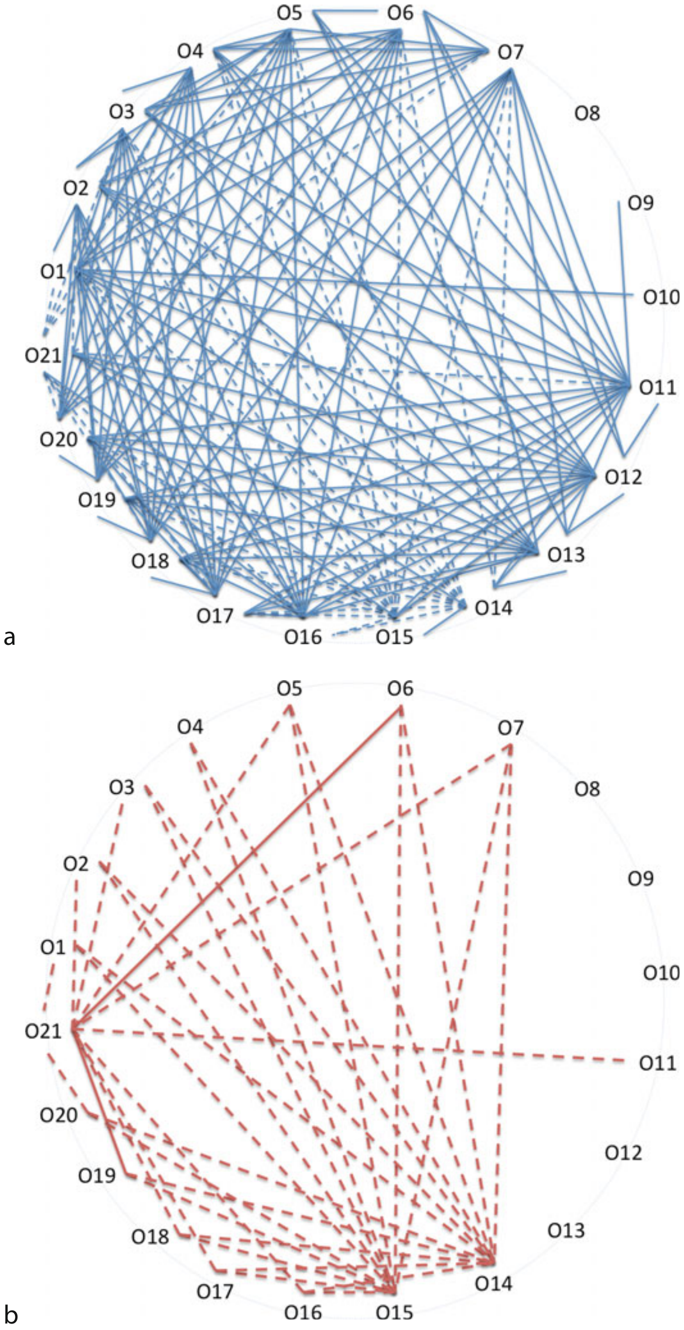


Fig. 16.13

Agreement (*top*) and disagreement (*bottom*) relations among the 21 test ontologies. *Plain lines* represent full disagreement/agreement (measures' values = 1). *Dashed lines* represent partial disagreement/agreement (measures' values greater than 0)

measures for all pairs of ontologies in this set. The results are shown in *Fig. 16.13* where ontologies are numbered according to their rank in Watson (valid on the 20/09/2009).

Analyzing these diagrams, it appears that there is a certain level of "coherence" in the results. In particular, homogeneous clusters can be built from the agreement and disagreement values: the ontologies O1, O2, O3, O4, O5, O6, O7, O11, O12, O13, O16, O17, O18, O19, and O20 all fully agree with each other and, at the same time, partially agree and disagree with O14 and O15. O14 and O15 also form a cluster since they agree with each other, and consistently disagree with the same set of ontologies (the reason being that O14 and O15 are the ontologies considering that SeaFood is a subclass of Meat, but agree on all the other related statements). O21 is also particular, since it disagrees with most of the ontologies of the first cluster, sometimes fully. Indeed, it also considers SeaFood to be a subclass of Meat, and additionally disagrees on several other statements with some of the other ontologies (e.g., it considers that tuna is a subclass of fish while several other ontologies consider tuna as an instance of fish). O8, O9, and O10 are particular since there is only a very small overlap between them and the other ontologies. For example, O9 only agrees with O11 that Vegan is a subclass of Vegetarian.

Another interesting piece of information that can be derived from the measures defined and from exploiting the collection of ontologies in Watson is the level to which particular statements are agreed with, that is, the level of consensus on a statement. Conversely, a related item of information concerns the level of controversy on the statement, that is, whether there is a clear-cut between agreement and disagreement. Here, a normalized mean was also used to measure the global agreement and disagreement of a statement *st* in a set of ontologies *R* (see details in [42]). From these two measures, consensus is defined as having a high level of certainty on whether ontologies in *R* agree or disagree with *st*. There is a high level of (positive consensus) if the overall agreement about this statement is high and the overall disagreement is low. Thus, the measure of consensus is computed in a set of ontologies *R* upon a statement *st* as follows:

$$consensus\,(st, R) = agreement\ \ (st, R) - disagreement\,(st, R)$$

The notion of controversy is then considered to be the inverse from the one of consensus: there is a high level of controversy on a given statement when there is no clear-cut between agreement and disagreement, that is, there is a low level of consensus. Therefore, the measure of controversy in a set of ontologies *R* upon a statement *st* can simply be computed in the following way:

$$controversy\,(st, R) = 1 - |consensus\,(st, R)|$$

To illustrate these measures, nine statements concerning the class *SeaFood* in Watson are considered. The results are summarized in *Table 16.1*.

As can be seen from these results, the first four statements are fully agreed with by ontologies in Watson, meaning that all the ontologies containing both entities of each statement express exactly the same relation as the one of the statement. The three next statements also have a very high level of agreement, and a very low level of disagreement. This is mainly due to a few ontologies containing the right entities, but not necessarily

Table 16.1

Consensus and controversy on statements concerning the *SeaFood* class in Watson

Statement	Consensus	Controversy
$<$ *SeaFood, disjointWith, Dessert* $>$	1.0	0.0
$<$ *Fowl, disjointWith, SeaFood* $>$	1.0	0.0
$<$ *Pasta, disjointWith, SeaFood* $>$	1.0	0.0
$<$ *SeaFood, subClassOf, EdibleThing* $>$	1.0	0.0
$<$ *ShellFish, subClassOf, SeaFood* $>$	0.89	0.109
$<$ *Fish, subClassOf, SeaFood* $>$	0.875	0.125
$<$ *SeaFood, disjointWith, Fruit* $>$	0.75	0.25
$<$ *Meat, disjointWith, SeaFood* $>$	0.53	0.46
$<$ *SeaFood, subClassOf, Meat* $>$	−0.719	0.281

describing any relation between them. Hence, there is a high level of consensus on these statements. Finally, the last two statements are the ones for which there is the highest level of controversy. The last one is by far the most disagreed with (which correlates with the high level of agreement of the other one contradicting it).

Another interesting example is the one of the statement $<$ *river, subClassOf, sea* $>$, which gives a high level of disagreement (0.766). The disagreement is not 1 in that case, because only very few ontologies express explicitly contradicting relations. However, in this case, the level of agreement is 0: There is no ontology to actually agree with this statement.

16.3 Related Resources

The previous sections give a detailed account of existing uses and applications of Semantic Web search engines, focusing in particular on two of the most prominent systems which are currently active. Due to the increase in the number of semantic documents made available online, and so to the need for search functionalities, a number of other systems have emerged recently from academic research (the list is deliberately restricted to systems that provide at least a freely accessible Web user interface for searching or querying semantic data):

Sindice (http://sindice.com/) is a *Semantic Web index* or *entity look-up service* that focuses on scaling to very large quantities of data. It provides keyword and URI-based search, structured query, and relies on some simple reasoning mechanisms for inverse-functional properties [26].

Falcons (http://iws.seu.edu.cn/services/falcons/) is a keyword-based semantic entity search engine. It provides a sophisticated Web interface that allows one to restrict the search according to recommended concepts or vocabularies [43].

SWSE (http://swse.deri.org/) is also a keyword-based entity search engine, but one that focuses on providing semantic information about the resulting entities rather than only links to the corresponding data sources [44]. Its collection is automatically gathered by crawlers. SWSE also provides a SPARQL endpoint enabling structured query on the entire collection.

Semantic Web Search (http://www.semanticwebsearch.com/) is also a semantic entity search engine based on keywords, but that allows one to restrict the search to particular types of entities (e.g., DOAP Projects) and provides structured queries.

OntoSelect (http://olp.dfki.de/ontoselect/) provides a browsable collection of ontologies that can be searched by looking at keywords in the title of the ontology or by providing a topic [45].

OntoSearch2 (http://www.ontosearch.org/) is a Semantic Web Search engine that allows for keyword search, formal queries, and fuzzy queries on a collection of manually submitted OWL ontologies. It relies on scalable reasoning capabilities based on a reduction of OWL ontologies in DL-Lite ontologies [46].

Sqore (http://ict.shinawatra.ac.th:8080/sqore) is a prototype search engine that allows for structured queries in the form of OWL descriptions [47]. Desired properties of entities to be found in ontologies are described as OWL entities and the engine searches for similar descriptions in its collection.

Finally, it is worth noticing that the issue of collecting semantic data from the Web has recently reached a broader scope, with the appearance of features within mainstream Web search engines exploiting structured data to improve the search experience and presentation. Indeed, Yahoo! SearchMonkey (http://developer.yahoo.com/searchmonkey/) crawls and indexes semantic information embedded in Web pages as RDFa (http://www.w3.org/TR/xhtml-rdfa-primer/) or microformats (http://microformats.org/), in order to provide enriched snippets describing the Web pages in the search results. Similarly, Google Rich Snippets (http://googlewebmastercentral.blogspot.com/2009/05/introducing-rich-snippets.html) makes use of collected semantic data using specific schemas in Web pages to add information to the presentation of results.

16.4 Conclusion and Future Directions

Semantic Web search engines are critical to the Semantic Web infrastructure. With the growth of Semantic Web data, applications and users, more and more research and development activities are being dedicated to building robust and scalable Semantic Web search engines. Most of the resulting systems are comparable in their structures and goals, but take different perspectives on the type of content they collect, on the task they support, and on the techniques they implement.

Developing such a system is a fascinating experience, touching on many different practical aspects of Semantic Web developments and including elements from other areas (information retrieval, interaction, databases, Web development, etc.), while integrating the tough constraint of reliability. But even more fascinating is the way Semantic Web search engines are used. They enable a new generation of applications that can benefit from a body of knowledge comparable to no other before. They allow users to explore this knowledge in efficient ways. They form a platform for researchers to study the Semantic Web and understand its content, its structure, and its evolution.

While Semantic Web search engines have gone a long way since the very first version of Swoogle (in 2004!), many research issues still need to be explored. The dynamic aspect of the Semantic Web will certainly become an important problem in the next few years and Semantic Web search engines will be required to come up with new solutions to deliver only valid, up-to-date knowledge. The implicit relationships that relate semantic documents should also be better explored, providing ways to really exploit the network of ontologies which is available online, in a currently very shallow form. Also, while the quality of online information is still a major issue, facilitating various levels of user contributions, from writing new ontologies to linking datasets and reviewing semantic information, seems an interesting direction for the future.

16.5 Cross-References

- KR and Reasoning on the Semantic Web: OWL
- KR and Reasoning on the Semantic Web: Web-scale Reasoning
- Ontologies and the Semantic Web
- Querying the Semantic Web: SPARQL
- Semantic Annotation and Retrieval: RDF
- Semantic Annotation and Retrieval: Web of Data
- Semantic Annotation and Retrieval: Web of Hypertext – RDFa and Microformats
- Semantic Web Architecture
- Storing the Semantic Web: Repositories

References

1. Sowa, J.F.: Conceptual graphs summary. In: Nagle, T.E., Nagle, J.A., Gerholz, L.L., Eklund, P.W. (eds.) Conceptual Structures: Current Research and Practice, pp. 3–51. Ellis Horwood, New York (1992) ISBN:0-13-175878-0
2. Hobbs, J.R., Ferguson, G., Allen, J., Fikes, R., Hayes, P., McDermott, D., Niles, I.: Adam Pease, Austin Tate, Mabry Tyson, Richard Waldinger. A daml ontology of time. http://www.cs.rochester.edu/~ferguson/daml/daml-time-nov2002.txt (2002)
3. Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: standard ontology for ubiquitous and pervasive applications. In: Proceedings of the First International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2004), Boston (2004)
4. Hayes, P.: RDF semantics. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/ (2004)
5. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL web ontology language semantics and abstract syntax. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/ (2004)
6. Ayers, D., Völkel, M.: Cool URIs for the semantic web, W3C Interest Group Note. http://www.w3.org/TR/cooluris/ (Sept 2010)
7. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. http://www.w3.org/TR/2006/WD-rdf-sparql-query-20060220/ (2006)
8. Bizer, C.: The emerging web of linked data. IEEE Intell. Syst. **24**, 87–92 (2009)
9. Bizer, C., Heath, T., Berners-Lee, T.: Linked data, the story so far. Int. J. Semantic Web Inf. Syst. **5**(3), 1–22 (2009)
10. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural language interfaces to databases: an introduction. Nat. Lang. Eng. **1**(1), 29–81 (1995)

11. Harth, A., Umbrich, J., Decker, S.: Multi-crawler: a pipelined architecture for crawling and indexing semantic web data. In: Proceedings of the Fifth International Semantic Web Conference (ISWC 2006), Athens, GA. Lecture Notes in Computer Science, vol. 4273, pp. 258–271. Springer, Berlin (2006)
12. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the semantic web. In: Proceedings of the Fourth International Semantic Web Conference (ISWC 2005), Galway. Lecture Notes in Computer Science, vol. 3729, pp. 156–170. Springer, Heidelberg (2005)
13. Alani, H., Brewster, C., Shadbolt, N.: Ranking ontologies with aktiverank. In: Proceedings of the Fifth International Semantic Web Conference (ISWC 2006), Athens, GA. Lecture Notes in Computer Science, vol. 4273, pp. 1–15. Springer, Berlin (2006)
14. d'Aquin, M., Euzenat, J., Le Duc, C., Lewen, H.: Sharing and reusing aligned ontologies with cupboard. In: Demo, Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP 2009), Los Angeles (2009)
15. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM 2004), Washington, DC, pp. 652–659 (2004)
16. Sherman, C.: Metacrawlers and metasearch engines. http://searchenginewatch.com/links/article.php/2156241 (last visited on March 2006) (2004)
17. Gulli, A., Signorini, A.: The indexable web is more than 11.5 billion pages. In: Proceedings of the 14th International World Wide Web Conference (WWW 2005) (poster paper), Chiba (2005)
18. Page, L., Brin, S., Motwani, R., Wino-grad, T.: The PageRank citation ranking: bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
19. d'Aquin, M., Sabou, M., Motta, E., Angeletou, S., Gridinoc, L., Lopez, V., Zablith, F.: What can be done with the semantic web? An overview of Watson-based applications. In: Proceedings of the Fifth Workshop on Semantic Web Applications and Perspectives (SWAP 2008), Rome (2008)
20. d'Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., Guidi, D.: Toward a new generation of semantic web applications. IEEE Intell. Syst. **23**(3), 20–28 (2008)
21. Angeletou, S., Sabou, M., Specia, L., Motta, E.: Bridging the gap between folksonomies and the semantic web: an experience report. In: Proceedings of the International Workshop on Bridging the Gap between Semantic Web and Web 2.0 at the Fourth European Semantic Web Conference (ESWC 2007), Innsbruck (2007)
22. Peroni, S., Motta, E., d'Aquin, M.: Identifying key concepts in an ontology through the integration of cognitive principles with statistical and topological measures. In: Proceedings of the Third Asian Semantic Web Conference (ASWC 2008), Bangkok. Lecture Notes in Computer Science, vol. 5367, pp. 242–256. Springer, Berlin (2009)
23. d'Aquin, M., Motta, E., Sabou, M., et al.: Towards a new generation of semantic web applications. IEEE Intell. Syst. **23**(3), 20–28 (2008)
24. Sabou, M., d'Aquin, M., Motta, E.: Exploring the semantic web as background knowledge for ontology matching. J. Data Semant. XI. Lecture Notes in Computer Science, vol. 5383, pp. 156–190, doi: 10.1007/978-3-540-92148-6_6 (2008)
25. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sigma: live views on the Web of Data. In: Demonstration at the Proceedings of the 19th World Wide Web Conference (WWW 2010), Raleigh (2010)
26. Tummarello, G., Oren, E., Delbru, R.: Sindice.com: weaving the open linked data. In: Proceedings of the Sixth International Semantic Web Conference and Second Asian Semantic Web Conference (ISWC/ASWC 2007), Busan. Lecture Notes in Computer Science, vol. 4825, pp. 547–560. Springer, Berlin (2007)
27. Finin, T.W., Sachs, J., Parr, C.S.: Finding data, knowledge, and answers on the semantic web. In: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2007), Key West, pp. 2–7. AAAI, Menlo Park (2007)
28. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In: Proceedings of the Second European Semantic Web Conference (ESWC 2005), Heraklion. Lecture Notes in Computer Science, vol. 3532, pp. 182–197. Springer, Berlin (2005)
29. Zablith, F.: Dynamic ontology evolution. In: Proceedings of the Seventh International Semantic Web Conference (ISWC 2008), Doctoral Consortium, Karlsruhe (2008)

30. Zablith, F., Sabou, M., d'Aquin, M., Motta, E.: Using background knowledge for ontology evolution. In: Proceedings of the Second International Workshop on Ontology Dynamics (IWOD 2008) Co-located with Seventh International Semantic Web Conference (ISWC 2008), Karlsruhe (2008)
31. Fernàndez, S., Berrueta, D., Shi, L., Labra, J.E., Ordóñez de Pablos, P.: Mailing lists and social semantic web. In: Patricia Ordonez de Pablos Miltiadis D. Lytras (ed.) Social Web Evolution: Integrating Semantic Applications and Web 2.0 Technologies. Social Computing: Concepts, Methodologies, Tools, and Applications Editor (s): Subhasish Dasgupta (George Washington University, USA), pp. 335–349 (2010)
32. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: a framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), Philadelphia (2002)
33. Lopez, V., Sabou, M., Motta, E.: PowerMap: mapping the real semantic web on the fly. In: Proceedings of the International Semantic Web Conference (ISWC 2006), Athens, GA. Lecture Notes in Computer Science, vol. 4273, pp. 414–427. Springer, Berlin (2006)
34. Angeletou, S., Sabou, M., Motta, E.: Semantically enriching folksonomies with FLOR. In: Proceedings of the First International Workshop on Collective Semantics: Collective Intelligence and the Semantic Web (CISWeb 2008), Tenerife (2008)
35. Ding, L., Finin, T.: Characterizing the semantic web on the web. In: Proceedings of the Fifth International Semantic Web Conference (ISWC 2006), Athens, GA. Lecture Notes in Computer Science, vol. 4273, pp. 242–257. Springer, Berlin (2006)
36. Lawrence, S., Lee Giles, C.: Accessibility of information on the web. Nature **400**, 107–109 (1999)
37. Bharat, K., Broder, A.: A technique for measuring the relative size and overlap of public web search engines. In: Proceedings of the Seventh International Conference on World Wide Web (WWW 1998), Brisbane, pp. 379–388 (1998)
38. d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., Motta, E.: Characterizing knowledge on the semantic web with Watson. In: Workshop on Evaluation of Ontologies and Ontology-Based Tools (EON 2007), Madrid (2007)
39. Wang, T.D., Parsia, B., Hendler, J.: A survey of the web ontology landscape. In: Proceedings of the Fifth International Semantic Web Conference, (ISWC 2006), Athens, GA. Lecture Notes in Computer Science, vol. 4273, pp. 682–694. Springer, Berlin (2006)
40. Allocca, C., d'Aquin, M., Motta, E.: Detecting different versions of ontologies in large ontology repositories. In: Proceedings of the Third International Workshop on Ontology Dynamics (IWOD 2009), Washington, DC. CEUR-WS Online Proceedings, vol. 519 (2009)
41. d'Aquin, M., Allocca, C., Motta, E.: A platform for semantic web studies. In: Web Science Conference (WebSci 2010), Poster Session, Raleigh (2010)
42. d'Aquin, M.: Formally measuring agreement and disagreement in ontologies. In: Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP 2009), Los Angeles (2009)
43. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: Proceedings of the 17th International Conference on World Wide Web (WWW 2008), Beijing, pp. 1101–1102. ACM, New York (2008)
44. Harth, A., Hogan, A., Delbru, R., Umbrich, J., O'Riain, S., Decker, S.: SWSE: Answers before links! In: Proceedings of the Semantic Web Challenge, CEUR Workshop Proceedings, vol. 295 (2007)
45. Buitelaar, P., Eigner, T., Declerck, T.: Ontose lect: a dynamic ontology library with support for ontology selection. In: Proceedings of the Demo Session at the Third International Semantic Web Conference (ISWC 2004), Hiroshima (2004)
46. Thomas, E., Pan, J.Z., Sleeman, D.H.: Ontosearch2: searching ontologies semantically. In: Proceedings of the Third International Workshop on OWL: Experiences and Directions (OWLED 2007), Innsbruck. CEUR Workshop Proceedings, vol. 258 (2007)
47. Ungrangsi, R., Anutariya, C., Wuwongse, V.: SQORE-based ontology retrieval system. In: Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA 2007), Regensburg. Lecture Notes in Computer Science, vol. 4653, pp. 720–729. Springer, Berlin (2007)
48. Myers, P., Espinosa, R., Parr, C.S., Jones, T., Hammond, G.S., Dewey, T. A.: The Animal Diversity Web (online). http://animaldiversity.org (2006). Accessed 5 Aug 2009