

Swoogle: A Search and Metadata Engine for the Semantic Web *

Tim Finin Yun Peng R. Scott Cost Joel Sachs Anupam Joshi
Pavan Reddivari Rong Pan Vishal Doshi Li Ding

University of Maryland Baltimore County, Baltimore MD 21250, USA
(finin,joshi)@cs.umbc.edu

ABSTRACT

Swoogle is a crawler-based indexing and retrieval system for the Semantic Web documents – i.e., RDF or OWL documents. It analyzes the documents it discovered to compute useful metadata properties and relationships between them. The documents are also indexed by using an information retrieval system which can use either character N-Gram or URIs as terms to find documents matching a user’s query or to compute the similarity among a set of documents. One of the interesting properties computed for each Semantic Web document is its rank – a measure of the document’s importance on the Semantic Web.

1. INTRODUCTION

Currently, the Semantic Web, in the form of RDF and OWL documents, is essentially a web universe parallel to the web of HTML documents. There is as yet no standard way for HTML (even XHTML) documents to embed RDF and OWL markup or to reference them in a standard way that carries meaning. Semantic Web documents reference one another as well as HTML documents in meaningful ways. This situation makes it appropriate to design and build specialized Internet search engines customized for Semantic Web Documents (SWDs).

Several interesting research issues are involved in this exercise: “what is the best way to index, digest and cache such systems?”, and “is it possible to create a meaningful *rank* measure that uses link semantics?”.

We describe a prototype internet search engine for SWDs encoded in RDF and OWL. The system is intended to support human users as well as software agents and services. At this stage, human users are expected to be semantic web researchers and developers who are interested in accessing,

*Partial research support provided by DARPA contract F30602-00-0591 and NSF award IIS-0326460.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM '04

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

exploring and querying a collection of metadata representing a significant fraction of the RDF and OWL documents found on the web. Software APIs will support programs that need to find SWDs matching certain descriptions, e.g., those containing certain terms, similar to other SWDs, using certain classes or properties, etc.

The system consists of a database that stores metadata about the SWDs, two distinct web crawlers that locate new and modified SWDs, components that compute useful document metadata, components to compute semantic relationships among the SWDs, an N-Gram based indexing and retrieval engine, a simple user interface for querying the system, and agent/web service APIs to provide useful services.

We describe an algorithm, *Ontology Rank*, inspired by the Page Rank algorithm [7, 5, 8], that is used to rank hits returned by the retrieval engine. This algorithm takes advantage of the fact that the graph formed by SWDs has a richer set of relations. In other word, the edges in this graph have explicit semantics. Some are defined or derivable from the RDF and OWL languages (e.g., imports, usesTerm, version, extends, etc.) and others by common ontologies (e.g., FOAF’s knows ¹). We will also present some preliminary results summarizing the characteristics of the portion of the semantic web that our system has crawled and analyzed.

We envision the following broad uses of Swoogle:

- *Finding appropriate ontologies.* Typically, an RDF editor allows a user to load an ontology, which she can then use to make assertions. But finding the right ontology to load is a problem, and the lack of an adequate solution has led to ontology proliferation. (Everybody just writes their own ontology). A user can query Swoogle for ontologies that contain specified terms anywhere in the document (including comments); for ontologies that contain specified terms as Classes or Properties; or for ontologies that are about a specified term (as determined by our IR engine). The ontologies returned are ranked according to the Ontology Rank algorithm, which seeks to capture the extent to which ontologies are being used by the community. We believe that this use of Swoogle will both ease the burden of marking up data, and contribute to the emergence of canonical ontologies.
- *Finding instance data.* The semantic web seeks to enable the integration of distributed information. But

¹<http://xmlns.com/foaf/0.1/>

first, the information must be found. A Swoogle user can query for all instance data about a specified class, or on a specified subject. The triples of the returned SWDs can then be loaded into a knowledge base for further querying.

- *Studying the structure of the semantic web.* The metadata computed by Swoogle will provide structural information about the semantic web. How connected is it? Which documents refer to an ontology? Which ontologies does a document refer to? What relationships (importing, using terms etc.) exist between two documents. Where is the graph most dense? etc.

2. SEMANTIC WEB DOCUMENTS

The semantic web languages based on RDF (e.g., RDFS², DAML+OIL³, and OWL⁴) allow one to make statements that define general terms (classes and properties), extend the definition of terms, create individuals, and to make assertions about the terms and individuals.

We call a document using semantic web languages a *Semantic Web Document* (SWD). We further restrict the scope of SWD to online documents that are accessible by web users and agents. Similar to a document in IR, a SWD is an atomic information exchange object in the Semantic Web.

Current practice favors the use of two kinds of documents which we will refer to as semantic web ontologies (SWOs) and semantic web databases (SWDBs). These correspond to what are called T-Boxes and A-Boxes in the description logic literature [1, 4]. We consider a document to be a SWO when a significant proportion of the statements it makes define new terms (e.g., new classes and properties) or extends the definition of terms defined in other SWDs by adding new properties or constraints. A document is considered as a SWDB when it does not define or extend a significant number of terms. A SWDB can introduce individuals and make assertions about them or make assertions about individuals defined in other SWDs.

For example, the SWD <http://xmlns.com/foaf/0.1/index.rdf> is considered a SWO in that its 466 statements (i.e. triples) define 12 classes and 51 properties but introduces no individuals. The SWD <http://umbc.edu/~finin/foaf.rdf> is considered to be a SWDB since it defines or extends no terms but defines three individuals and makes statements about them.

Besides the two extremes, some SWDs may well be intended to be both an ontology that defines a set of terms to be used by others as well as a useful database of information about a set of individuals. Even a document that is intended as an ontology (e.g., it defines a set of terms that can be used by others) might define individuals as part of the ontology. Similarly, a document that is intended as defining a set of individuals might introduce some new terms in order to make it easier to describe the individuals rather than to make them available for others to use,⁵ e.g. <http://www.daml.ri.cmu.edu/ont/USCity.daml>, <http://reliant.tekknowledge.com/DAML/Government.owl>.

²<http://www.w3.org/TR/rdf-schema/>

³<http://www.w3.org/TR/daml+oil-reference>

⁴<http://www.w3.org/TR/owl-semantics/>

⁵Programming languages introduce the notion of a modules, importing and exporting to make these intentions explicit

3. SWOOGLE ARCHITECTURE

As shown in figure 1, Swoogle's architecture can be broken into four major components: SWD discovery, metadata creation, data analysis, and interface. This architecture is data centric and extensible: different components work on different tasks independently.

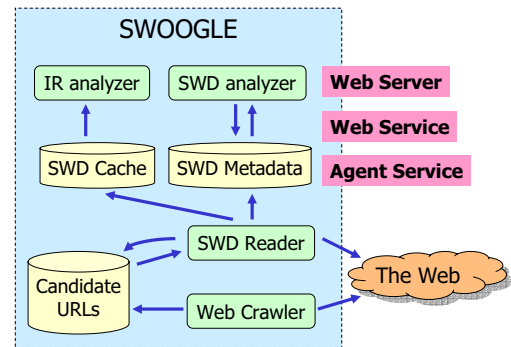


Figure 1: The architecture of Swoogle

The *SWD discovery* component is responsible to discover the potential SWDs throughout the Web and keep up-to-date information about SWDs.

The *metadata creation* component caches a snapshot of a SWD and generates objective metadata about SWDs in both syntax level and semantic level.

The *data analysis* component uses the cached SWDs and the created metadata to derive analytical reports, such as classification of SWO and SWDB, rank of SWDs, and the IR index of SWDs.

The *interface* component focuses on providing data service to the Semantic Web community. We have implemented Web interface at <http://www.swoogle.org>, and we are working on making Swoogle a web service.

In the following section, we will elaborate the four components respectively.

4. FINDING SWDS

It is interesting, but also hard, to estimate the scale of the Semantic Web, which can be characterized by the number of SWDs available on the Web. Although the absolute number of SWDs is likely nontrivial, it is quite small in comparison with whole collection of web documents. Google for instance has crawled and indexed 4,285,199,774 pages by May 25, 2004. Finding SWDs to analyze and index is in itself an interesting engineering challenge, in part because the Semantic Web is not very well connected. As we will see later, most SWDs do not have many inlinks. We design a two phased solution in Swoogle to overcome this problem.

We start by querying Google for files with suffixes such as “.rdf”, “.owl”, “.nt”, and “.n3”. But our statistics shows that this list is not complete, nor are all files with suffixes in this list SWDs.

We developed a *Google crawler* to search URLs using Google Web Service. Basically, we search for files with specific extensions (as shown in table 1). The only issue with Google is that the API does not return more than 1000 results for any query. Therefore, we append some more keywords to split one big query into several queries whose re-

query string	number of pages
rdf	5,230,000
filetype:rdf rdf	246,000
filetype:rss rdf	13,800
filetype:daml rdf	4,360
filetype:n3 rdf	2,630
filetype:owl rdf	1,310
filetype:nt rdf	415
filetype:rdfs rdf	304

Table 1: Google search results (May 25,2004)

sults can be combined. Using this, we can get as many as 20k candidate SWDs a time. Moreover, we repeat this process, since Google may have indexed new URLs, altering the original PageRanks and returning a new set of results. We use some simple heuristics, such as file extension and content pattern, to filter out the obviously irrelevant documents so as to reduce search complexity and improve search precision. In the second phase, we crawl all the outlinks from these candidate SWDs recursively. We developed a Jena⁶ based *Swoogle crawler* that can analyze the content of SWD as well as discover candidate SWDs.

The crawler maintains two URL lists: one which contains the candidate URLs to be visited and the other is a list of discovered SWDs. It not only schedules visiting newly discovered URLs, but also revisits URLs to check for updates.

Swoogle also provides a web interface where registered users can submit URLs which is believed to be SWDs. We also allow users to submit a URL of a web directory under which many SWDs may be present, e.g. <http://daml.umbc.edu/ontologies/>. We have built a *focused crawler* which only crawls all URLs relative to the given base URL.

It is nontrivial to determine if a URL encountered in crawling leads to an SWD. We use heuristic to filter out those URLs which are obviously not SWDs, e.g. jpg, html, and gif. Another heuristic uses the semantics of ontology. In a FOAF file for instance, the value of `rdfs:seeAlso` will lead to another FOAF file, which is a SWD.

5. SWD METADATA

The SWD metadata is collected to enhance the efficiency and effectiveness of SWD search. It is derived from the content of SWD and relations between SWDs. Currently, we identify three categories of metadata: (i) basic metadata, which considers the syntactic and semantic features of a SWD, (ii) relations, which consider the explicit semantics between individual SWDs, and (iii) analysis results such as SWO/SWDB classification, and SWD ranking. The first two categories are objective information about SWDs, and we will discuss them in the rest of this section. The third category is derived as subjective evaluation, and the detailed discussion is in section 6.

5.1 Basic metadata

Swoogle captures the metadata of a SWD by analyzing it using Jena. We identify three groups of properties directly related to a SWD: language features, RDF statistics and ontology annotations.

⁶<http://jena.sourceforge.net/>

Language features refers to the properties describing the syntax and semantics of a SWD. They are derived from parsing results. Swoogle is collecting the following properties:

1. *Encoding*. It indicates the syntactic encoding of a SWD. We currently look at two types of encodings, namely RDF/XML and N3.
2. *Language*. It indicates the semantic language used by a SWD. We currently look at four types of languages, namely OWL, DAML+OIL, RDFS, and RDF.
3. *OWL Species*. It is specific to only OWL documents. It indicates the OWL species of a OWL document, namely OWL-LITE, OWL-DL, and OWL-FULL.

RDF statistics refers to the properties summarizing the node distribution of the RDF graph in a SWD. We focus on how SWDs define new classes, properties and individuals. In an RDF graph, a node is identified as a *class* iff it is not an anonymous node and it is an instance of `rdfs:Class`; similarly, a node is a *property* iff it is not an anonymous node and it is an instance of `rdf:Property`; an *individual* is a node which is an instance of any user defined class. By parsing a SWD *foo* into a RDF graph, we may get the RDF statistics about *foo*, let $C(foo)$ be the set of classes defined in *foo*, $P(foo)$ be the set of properties defined in *foo*, and $I(foo)$ be the individuals defined in *foo*. The *ontology-ratio* $R(foo)$ of *foo* is calculated by equation 1. The value of ontology-ratio ranges from 0 to 1. 0 implies that *foo* is a pure SWDB while 1 implies that *foo* is a pure SWO.

$$R(foo) = \frac{C(foo) + P(foo)}{C(foo) + P(foo) + I(foo)} \quad (1)$$

Ontology annotations are the properties that describe an ontology. In practice, when a SWD is written in OWL and has an instance of `OWL:Ontology`, we simply copy the corresponding property values. We currently record the following properties:

1. *Number of imports*. It captures the number of SWOs imported by this SWD.
2. *Label*.
3. *Comment*.
4. *Version Info*.

5.2 Relations between SWDs

We assume that the relationship between any two given SWDs has to be one of the following:

- IM** Imports (When source ontology imports destination ontology)
- EX** Extends (When source ontology extends destination ontology)
- TM** Uses Term (When source ontology uses one or more terms (classes or properties) from destination ontology)
- IN** Uses Individual (When source ontology uses one or more individuals from destination ontology)
- PV** Prior Version (When destination ontology is a prior version of source ontology)

Type	Classes and Properties
IM	owl:imports, daml:imports
EX	rdfs:subClassOf, rdfs:subPropertyOf, owl:disjointWith, owl:equivalentClass, owl:equivalentProperty, owl:complementOf, owl:inverseOf, owl:intersectionOf, owl:unionOf daml:sameClassAs, daml:samePropertyAs, daml:inverseOf, daml:disjointWith daml:complementOf, daml:unionOf daml:disjointUnionOf, daml:intersectionOf
PV	owl:priorVersion
CPV	owl:DeprecatedProperty, owl:DeprecatedClass, owl:backwardCompatibleWith
IPV	owl:incompatibleWith

Table 2: OWL, DAML+OIL and RDFS classes and properties related to SWD relationships

CPV When destination ontology is a prior version of source ontology, and is compatible with it

IPV When destination ontology is a prior version of source ontology, and is incompatible with it.

Source ontology is the input file currently being processed while destination ontology is the value of the rdf:object in a triple. The SWD relationships are computed by looking at the predicate values of the triples in a SWD. For OWL, DAML+OIL and RDFS the predicates that identify the above relationships are described in Table 2:

Using Jena, we can check if the value of a predicate in a parsed document matches an entry in table 2. If so, we enter the appropriate relationship into the database.

6. RANK SWDS

PageRank [7, 3] is a measure popularized by Google to evaluate the relative importance of web documents. Given a document A, As PageRank is computed as:

$$PR(A) = (1 - d) + d(PR(T_1)/C(T_1) + \dots + PR(T_n)/C(T_n)) \quad (2)$$

where, T_1, \dots, T_n are web documents that link to A. $C(T_i)$ is the total outlinks of T_i . d is damping factor, which equals 0.85 in practice. The basic idea behind this is that PageRank measures the probability that a random surfer will visit a page. The equation captures the probability that a user can arrive at a given page by following one of the links pointing to it, or simply by directly addressing it – the second and the first terms in Equation 2 respectively.

Unfortunately, this model is not appropriate for the Semantic Web, where links have semantics, and thus following any outgoing link from a page is no longer equiprobable. We assume that a SWD has, in general, sets of other SWDs that (i) import it; (ii) use some of its terms without importing it; (iii) extends the definitions about terms it defines; and (iv) make assertions about the individuals it defines.

A surfer in the Semantic Web should not be purely random, but rather rational. For instance, if an SWD imports some SWO, then clearly that link must be followed – the present SWD would not make any sense without the SWO it imports. Similarly, the surfer is more likely to visit an SWD from which current SWD extends a class rather than

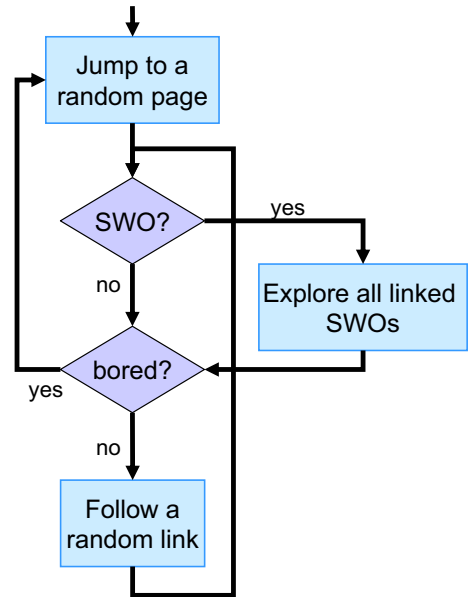


Figure 2: Rational Random Surfer

an SWD from which it refers an individual, because the former provides more useful information for him/her to understand the present document. Amongst links that represent referring of individuals from other SWDs, the number of individuals referred influences the probability of going to that document, for example, a document with one individual referred is less likely to be visited than the one from which a hundred have been referred. As such, we assign different types of SWD relations different weights to model their probability of being explored:

$$\begin{aligned}
 rawPR(a) &= (1 - d) + d \sum_{x \in L(a)} rawPR(x) \frac{f(x,a)}{f(x)} \\
 f(x,a) &= \sum_{l \in links(x,a)} weight(l) \\
 f(x) &= \sum_{a \in T(x)} f(x,a)
 \end{aligned} \quad (3)$$

where $L(a)$ stands for all SWDs that links to a , $T(x)$ stands for all SWDs that x links to.

SWDB:

$$SWI : PR(a) = rawPR(a) \quad (4)$$

SWO:

$$SWO : PR(a) = \sum_{x \in TC(a)} rawPR(x) \quad (5)$$

where $TC(a)$ stands for a 's transitive closure of SWOs.

Like in PageRank – we retain a finite probability that even our rational surfer will show some randomness, and directly visit some document. However, when deciding on which link to follow from a given page, the probability is given by $\frac{f(x,a)}{f(x)}$, where x is the current SWDB, a is the SWD that x links to, $f(x,a)$ is the sum of all link weights from x to a , and $f(x)$ is the sum of the weights of all outlinks from x . We call this approach the “Rational Random Surfer”, whose control flow is shown in figure 2.

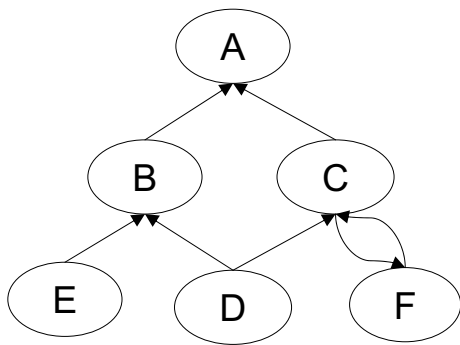


Figure 3: SWD rank flow

In figure 3, A, B, C, D, E and F are all SWOs. If the probability for a rational random surfer to visit each document from a SWDB is 0.0001, the probability that she visits B and C is $0.0001+(0.0001+0.0001)=0.0003$, the probability that she visits A is $0.0001+(0.0001*5)=0.0006$, and the probability that she visits F is $0.0001+0.0001=0.0002$.

The SWDs with the highest ranks are detailed in Table 3.

7. INDEXING AND RETRIEVAL OF SWDS

Central to a semantic web search engine is the problem of searching collections of marked documents and retrieve relevant text. While there is significant semantic information encoded in marked documents, reasoning over large collections of documents can be expensive. Traditional IR techniques have the advantage of being faster, while taking a somewhat more coarse view of the text. They can thus quickly retrieve a set of SWDs that deal with a topic based on text similarity alone.

In addition to the efficiency, there are a number of reasons why one would want to apply IR techniques to this problem. For one thing, documents are not entirely markup. We would like to be able to apply search to both the structured and unstructured components of a document. Related to this point, it is conceivable that there will be some text documents that contain embedded markup. In addition, we may want to make our documents available to commonly used search engines, such as Google. This implies that the documents must be transformed into a form that a standard IR engine can understand and manipulate. IR techniques also have some value characteristics, including well researched methods for ranking matches, computing similarity between documents, and employing relevance feedback.

There has been work [9, 6] demonstrating that such techniques can be made to work with marked text, and that they can be made to leverage some of the semantic information encoded.

Traditional IR techniques look at a document as either a collection of words or N-Gram. An N-Gram is an n-character segment of the text which spans inter-word boundaries. The N-Gram approach is typically employed by sliding a window of n-characters along the text, and taking a sample at each one character step. The use of N-Grams can result in a larger vocabulary, as single words can contain multiple N-Grams. One advantage to this approach is that inter-word relationships are preserved, where they are typically not in word based approaches. N-Grams are also

known to be somewhat resistant to errors.

The use of N-Gram is particularly important to this approach because of the treatment of URIs as terms. Given a set of keywords defining a search, we may want to match documents that have URIs containing those keywords. For example, consider a search for ontologies for “time”. The search keywords might be *time temporal interval point before after during day month year eventually calendar clock durations end begin zone*. Candidate matches might include documents containing URIs such as:

```

http://foo.com/timeont.owl#timeInterval
http://foo.com/timeont.owl#CalendarClockInterval
http://purl.org/upper/temporal/t13.owl#timeThing
  
```

Clearly, exact matching based on words only would miss these documents (based on the URIs given). However, N-Grams would find a number of matches.

It is also possible, however, to use a word based approach. Bear in mind that ontologies define vocabularies. In OWL, URIs of classes and properties are the words. We can take an SWD, reduce it to triples, extract the URIs (with duplicates), discard URIs for blank nodes, hash each URI to a token, and index the document. This algorithm is the basis for the Swangling approach. Swangling is defined as:

1. the conversion of an RDF triple into one or more IR indexing terms, or
2. the processing of a document or query so that its content baring markup will be indexed by an IR system.

As in most IR approaches, queries are processed in the same manner as documents. URIs are hashed using the MD5Hash algorithm. The effect of Swangling is that a SW triple is converted into seven word like terms; one for each non-empty subset of the three components with the missing elements replaced by the special *don't care* URI, and terms generated by a hashing function. The original document is then augmented by the addition of triples containing the swangle terms. The benefit here is that the document can be indexed by a conventional engine, such as Google.

Here, we present an example of a swangled SW triple:

```

<rdf:RDF
  xmlns:s="http://swoogle.umbc.edu/ontologies/swangle.owl#"
</rdf>

<s:SwangledTriple>
  <s:swangledText>N656WNTZ36KQ5PX6RFUGVKQ63A</s:swangledText>
  <rdfs:comment>Swangled text for
  [http://www.xfront.com/owl/ontologies/camera/#Camera,
  http://www.w3.org/2000/01/rdf-schema#subClassOf,
  http://www.xfront.com/owl/ontologies/camera/#PurchaseableItem]
  </rdfs:comment>
  <s:swangledText>M6IMWPIH4YQI4IMGZYBGPYKEI</s:swangledText>
  <s:swangledText>H02H3FOPAEM53AQIZ6YVPPQ2XI</s:swangledText>
  <s:swangledText>2AQEUJOYPMXWKHZTENIJS6PQ6M</s:swangledText>
  <s:swangledText>IIVQRXOAYRH6GGRZDFXKKEB4PY</s:swangledText>
  <s:swangledText>75Q5Z3BYAKRPLZDLFNS5KKMTOY</s:swangledText>
  <s:swangledText>2FQ2YI7SNJ7OMXOXIDEE2W0ZU</s:swangledText>
</s:SwangledTriple>
  
```

For this project, we have used the Sire IR engine, a custom engine we built for the Carrot2 distributed IR system [2]. Sire can be made to use either n-grams or words, and employs a TF/IDF model with a standard cosine similarity metric. Sire has been adapted to accommodate the processing of swangled URI's as terms in the text.

Rank	URL	Value
1	http://www.w3.org/1999/02/22-rdf-syntax-ns	2845.97
2	http://www.w3.org/2000/01/rdf-schema	2814.21
3	http://www.daml.org/2001/03/daml+oil	311.65
4	http://www.w3.org/2002/07/owl	192.18
5	http://www.w3.org/2000/10/rdf-tests/rdfcore/testSchema	59.82
6	http://www.w3.org/2002/03owl/testOntology	22.50
7	http://ontology.ihmc.us/Entity.owl	21.2825
8	http://www.w3.org/2001/XMLSchema	17.45
9	http://www.daml.org/2000/12/daml+oil	10.44
10	http://www.daml.org/2000/10/daml-ont	8.88
11	http://ontology.ihmc.us/Group.owl	7.67
12	http://ontology.ihmc.us/Actor.owl	6.83

Table 3: SWDs with highest ranks

8. CURRENT STATUS AND FUTURE WORK

Swoogle is an ongoing project undergoing constant development. This paper describes the features in version 1. Figure 4 shows the welcome interface of Swoogle.

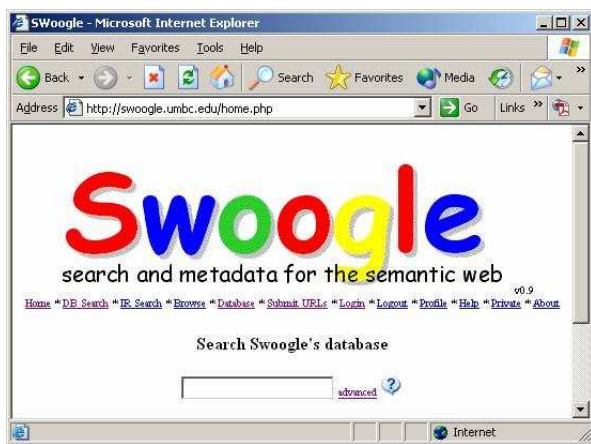


Figure 4: Swoogle interface

A general user can query with keywords, and the SWDs matching those keywords are returned in ranked order.

We observe that our ranking algorithm ranks SWOs higher than SWDBs; thus, SWOs using those query terms will be returned before SWDBs using those terms. The highest ranked SWDs typically are the base ontologies that define the semantic web languages, such as the RDF or OWL definitions, which all SWDs must import.

For advanced users, an “advanced” search interface is provided (figure 5), which essentially allows them to fill in the constraints to a general SQL query on the underlying database.

The user can query using keywords, content based constraints (type of SWD, number of classes/properties/individuals), language and encoding based constraints (N3 vs XML), and/or the Rank of the document. The result is shown in figure 6.

At present, the metadata are stored in a MySQL database, and indexes about 11000 SWDs. We anticipate that in future versions, we will need to migrate to some commercial databases in order to scale to indexing hundreds of thou-

sands of SWDs.

In our database, 13.29 percent SWDs are classified as SWOs, and others as SWDBs with the threshold as 0.8. We find about half of total SWDs have rank of 0.15, which means they are not referred by any other SWDs. Also, the mean of ranks is 0.8376, which implies that the SWDs are poorly connected. Figure 7 shows the log-log plot of SWD size distribution. The size of SWD is measured by the number of triples present in a SWD.

Current Swoogle (version 1) does not cache the SWDs visited. In the present version of Swoogle we have the focused crawler running. So, a user can give a directory path and the focused crawler will crawl through it and add all potential SWDs to the Swoogle crawler queue.

9. CONCLUSIONS

Current web search engines such as Google and AlltheWeb do not work well with documents encoded in the semantic web languages RDF and OWL. These retrieval systems are designed to work with natural languages and expect documents to contain unstructured text composed of words. They do a poor job of tokenizing semantic web documents and do not understand conventions such as those involving XML namespace. Moreover, they do not understand the structural information encoded in the documents and are thus unable to take advantage of it.

Semantic web researchers need search and retrieval systems today to help them find and analyze semantic web documents on the web. These systems can be used to support the tools being developed by researchers – such as annotation editors – as well as software agents whose knowledge comes from the semantic web.

We have described a prototype crawler-based indexing and retrieval system for the semantic web, i.e., web documents written in RDF or OWL. It analyzes the documents it discovers to compute useful metadata properties and relationships between them. The documents are also indexed using an information retrieval system which can use either character N-Grams or URIs as terms to find documents matching a user’s query or compute the similarity among a set of documents. One of the interesting properties computed for each semantic web document is its rank – a measure of the documents importance on the semantic web.

The current version of our system has discovered and analyzed over 11,000 semantic web documents. A second ver-

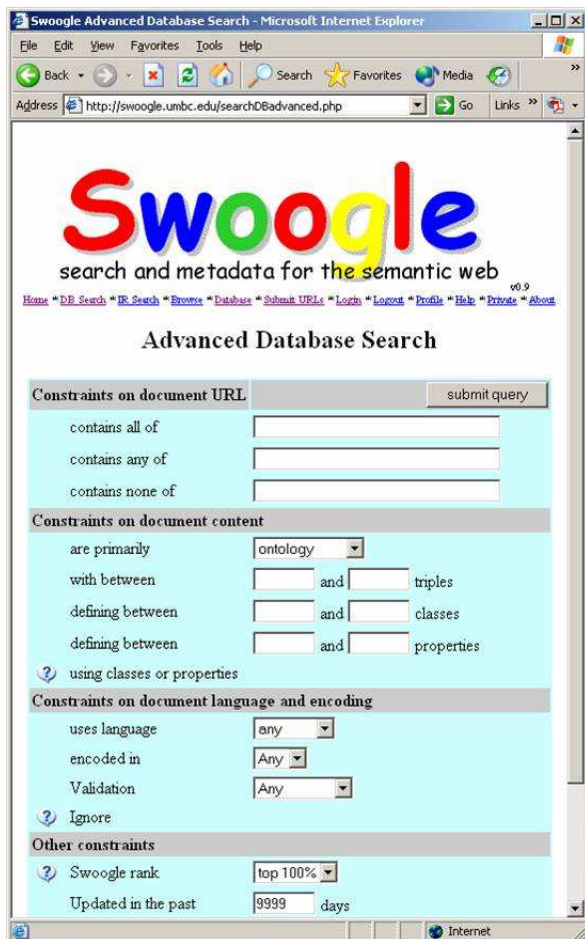


Figure 5: Swoogle advanced query

sion has been designed and partially implemented that will also store meta data on classes and properties and is designed to support millions of documents.

10. REFERENCES

- [1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [2] R. S. Cost, S. Kallurkar, H. Majithia, C. Nicholas, and Y. Shi. Integrating distributed information sources with carrot ii. In *Proceedings of the 6th International Workshop on Cooperative Information Agents VI*, pages 194–201. Springer-Verlag, 2002.
- [3] T. Haveliwala. Efficient computation of pageRank. Technical Report 1999-31, 1999.
- [4] I. Horrocks. DAML+OIL: A description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9, 2002.
- [5] M. Lifantsev. Rank computation methods for Web documents. Technical Report TR-76, ECSL, Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY, November 1999.
- [6] J. Mayfield and T. Finin. Information retrieval on the

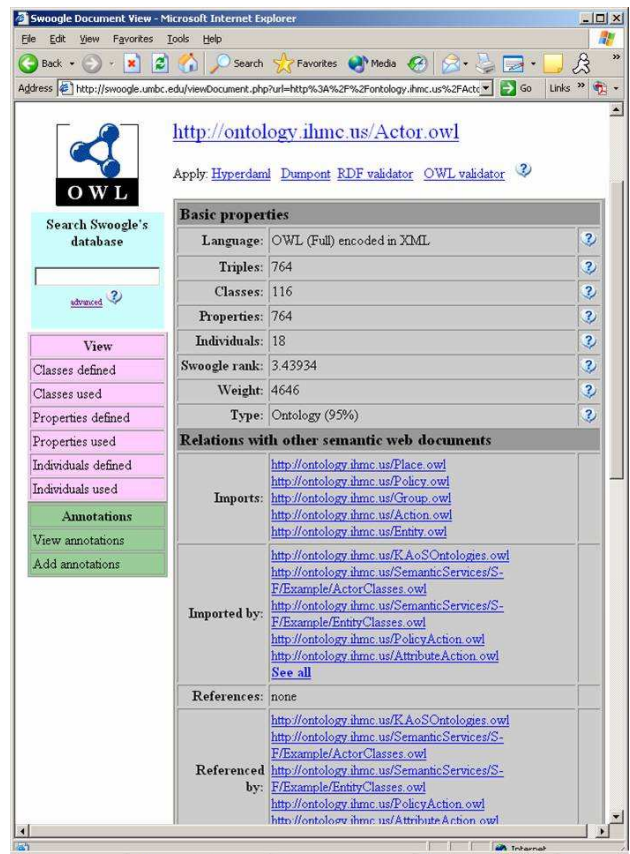


Figure 6: Swoogle query result

semantic web: Integrating inference and retrieval. In *Proceedings of the SIGIR 2003 Semantic Web Workshop*, 2003.

- [7] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [8] I. Rogers. The google pagerank algorithm and how it works. <http://www.iprcom.com/papers/pagerank/>, May 2002.
- [9] U. Shah, T. Finin, and A. Joshi. Information retrieval on the semantic web. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 461–468. ACM Press, 2002.

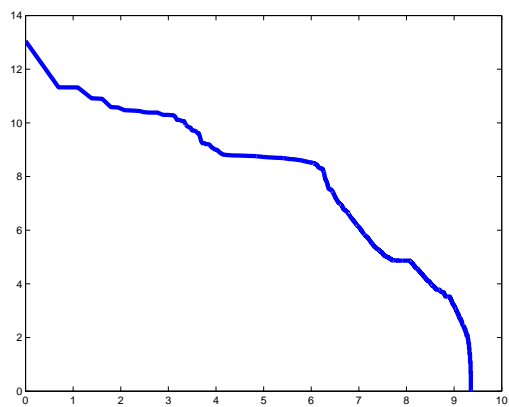


Figure 7: The Log-Log plot of SWD size distribution