Information Retrieval and the Semantic Web

Article · January 2005

DOI: 10.1109/HICSS.2005.319 · Source: DBLP

5 authors, including:

Tim Finin
University of Maryland, Baltimore County
570 PUBLICATIONS **29,491** CITATIONS

SEE PROFILE

Anupam Joshi
University of Maryland, Baltimore County
505 PUBLICATIONS **19,224** CITATIONS

SEE PROFILE

Richard Scott Cost
Johns Hopkins University
47 PUBLICATIONS **2,059** CITATIONS

SEE PROFILE

Information Retrieval and the Semantic Web

Tim Finin[1], James Mayfield[2], Anupam Joshi[1], R. Scott Cost[2] and Clay Fink[2]

University of Maryland,[1]
Baltimore County
Baltimore MD 21250 USA

The Johns Hopkins University[2]
Applied Physics Laboratory
Laurel MD 20723 USA

Abstract

Information retrieval technology has been central to the success of the Web. For semantic web documents or annotations to have an impact, they will have to be compatible with Web based indexing and retrieval technology. We discuss some of the underlying problems and issues central to extending information retrieval systems to handle annotations in semantic web languages. We also describe three prototype systems that we have implemented to explore these ideas.

1. Introduction

Information retrieval technology has been central to the success of the Web. Web based indexing and search systems such as Google and Yahoo have profoundly changed the way we access information. For the semantic web technologies [4][5] to have an impact, they will have to be compatible with Web search engines and information retrieval technology in general. We discuss several approaches to using information retrieval systems with both semantic web documents and with text documents that have semantic web annotations.

One vision of the Semantic Web is that it will be much like the Web we know today, except that documents will be enriched by annotations in machine understandable markup. These annotations will provide metadata about the documents as well as machine interpretable statements capturing some of the meaning of the documents' content. We describe initial experiments that demonstrate how existing IR systems can be coaxed into supporting this scenario using a technique we call *swangling* to encode RDF triples as word-like terms.

In an alternate vision, semantic web content will exist in separate documents that reference and describe the content of conventional web documents. Here too it may be desirable to use conventional systems such as Google to index and retrieve these documents. We discuss how the swangling technique can also be used to add assertions to RDF documents in a way that is compatible with many standard search engines.

A final approach to using IR engines for SWD documents is to build custom indexing and retrieval engines specifically designed to work with semantic web documents as opposed to conventional ones. We describe Swoogle, a prototype crawler-based search engines for RDF documents. This system allows users to retrieve indexed RDF documents based on the RDF classes and properties they use and also uses the Haircut information retrieval engine to retrieve documents using character-based n-grams.

The next section will motivate the ability to index and search for documents consisting of or annotated with semantic web content. Section Three will lay out the landscape of possible ways to adapt information retrieval systems to the Semantic Web and Section Four will describe three different prototype systems we have built to explore the problem. The fifth section summarizes this work and speculates on what the future may bring.

2. Motivation

The Semantic Web has lived its infancy as a clearly delineated body of Web documents. That is, by and large researchers working on aspects of the Semantic Web knew where the appropriate ontologies resided and tracked them using explicit URLs. When the desired Semantic Web document was not at hand, one was more likely to use a telephone to find it than a search engine. This closed world assumption was natural when a handful of researchers were developing DAML 0.5 ontologies, but is untenable if the Semantic Web is to live up to its name. Yet simple support for search over Semantic Web documents, while valuable, represents only a small piece of the benefits that will accrue if search and inference are considered together. We believe that Semantic Web inference can improve traditional text search, and that text search can be used to facilitate or augment Semantic Web inference. Several difficulties, listed below, stand in the way of this vision.

Current Web search techniques are not directly suited to indexing and retrieval of semantic markup. Most search engines use words or word variants as indexing terms. When a document written using some flavor of SGML is indexed, the markup is simply ignored by many search engines. Because the Semantic Web is expressed entirely as markup, it is thus invisible to them. Even when search engines detect and index embedded markup,

they do not process the markup in a way that allows the markup to be used during the search, or even in a way that can distinguish between markup and other text.

Current Web search techniques cannot use semantic markup to improve text retrieval. Web search engines typically rely on simple term statistics to identify documents that are most relevant to a query. One might consider techniques such as thesaurus expansion or blind relevance feedback to be integration of inference into the retrieval process, but such inference is simple compared with what is possible using semantic markup. One would like the presence of semantic markup in either the query or the documents retrieved to be exploitable during search to improve that search.

Likewise, text is not useful during inference. To the extent that it is possible to automatically convert text to a semantic representation, such resulting representations can be used during inference. However, semantic interpretation is difficult at best, and unsolved in the general case. We would like a way to exploit relevant text during inference, without needing to analyze the semantics of that text.

There is no current standard for creating or manipulating documents that contain both HTML text and semantic markup. There are two prime candidates for such hybrid documents. First, semantic markup might be embedded directly in an HTML page. Unfortunately, while we call approaches like RDF and OWL semantic *markup*, they are typically used not as markup but rather as stand-alone knowledge representation languages that are not directly tied to text. Furthermore, embedding RDF-based markup in HTML is non-compliant with HTML standards up to and including HTML 4.0. This issue is currently under study by a W3C task force [23].

The second way to bind HTML to semantic markup is to create a pair of documents, one containing HTML, the other containing the corresponding semantic markup. The two files are bound by placing in each a pointer to the URI of the other, either by URI naming convention, or by concurrent retrieval (*i.e.*, as part of a single transaction). While this method makes it difficult to associate semantic markup with specific components of the HTML page, it is possible to implement using today's standards. Whichever approach is taken to binding semantic markup to HTML, the current lack of a standard has made it difficult to exploit the relationship between the two.

One of the stated objectives of the semantic web is to enhance the ability of both people and software agents to find documents, information and answers to queries on the Web. While there has been some research on information retrieval techniques applied to documents with markup [1][2][3][7][13], combining retrieval with ontology browsing [9], the role of explicit ontologies in information retrieval tasks [19], and on question answering as a retrieval task [18], much of it can be seen as incremental extensions to familiar paradigms. Our goal is more ambitious and offers, we think, a new paradigm for information retrieval that mixes and interleaves search, retrieval and understanding.

To explore the tight integration of search and inference, we propose a framework designed to meet the following desiderata:

- The framework must support both retrieval-driven and inference-driven processing.
- Retrieval must be able to use words, semantic markup, or both as indexing terms.
- Web search must rely on today's broad coverage, text-based retrieval engines.
- Inference and retrieval should be tightly coupled; improvements in retrieval should lead to improvements in inference, while improvements in inference

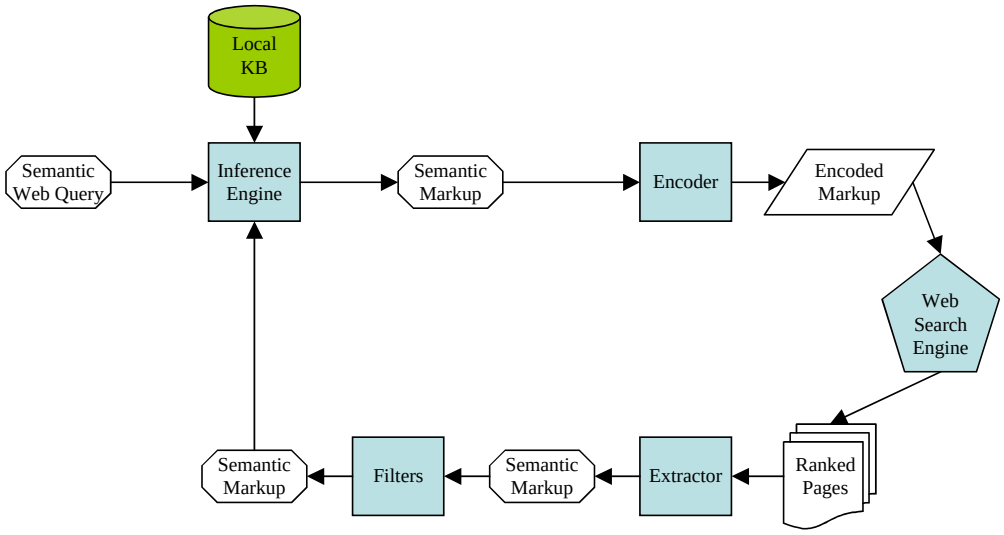


Figure 1. Integration of inference and retrieval over semantic markup. Arrows represent data flow.

should lead to improvements in retrieval.

In the following subsections, we first describe the portions of the framework that use semantic markup, then show how text processing can be mixed in to increase system capabilities and improve performance.

2.1 Processing of Semantic Markup

Imagine we are concerned only with retrieval and inference over semantic markup. We would like the ability to operate some sort of inference engine, to identify facts and rules needed by the inference engine to reach its desired conclusions, to search the Semantic Web for such facts and rules, and to incorporate the results of the search into the inference process. Figure 1 shows the basic architecture of such a system.

Input to the system is some sort of Semantic Web query. If the user's goal is retrieval, this might simply be semantic markup encoding the concepts being sought (*e.g.*, using XML-QL [10] or XIRQL [15]). Alternatively, if the goal is inference, the query might be a statement the system is to prove. In either case, the query is submitted to the inference engine. For retrieval, the inference engine may choose to perform limited forward chaining on the input (as a text retrieval engine might perform thesaurus expansion). For proof, the inference engine will generate a partial proof tree (or more accurately, one in a sequence of partial proof trees), using its local knowledge base to the extent possible. The inference engine produces a description of the semantic markup to be sought on the Web.

Because we want to use a traditional Web search engine for the retrieval, we cannot simply use the output of the inference engine as a search query. Rather, we must first encode the semantic markup query as a text query that will be recognized by a search engine. We call this process *swangling*, for 'Semantic Web mangling.'[1] Technical details about swangling, and its application to Web pages prior to indexing, are discussed further below in Section 4. The result is a bag of words, recognizable as indexing terms by the target Web search engine(s), that characterize the desired markup.

The query is submitted to one or more Web search engines. The result will be a ranked list of Web pages, which either contain semantic markup themselves, or refer to companion pages that do. Some number of these pages must be scraped to retrieve their semantic markup. Control over how many pages to scrape, and over whether to scrape additional pages or to issue a new Web query, resides with the inference engine.

[1] Mangling is the technical term for a technique used in C++ and other object-oriented compilers in which the types of a method's arguments and return value are encoded in the internal function name.

Only some of the semantic markup retrieved through this process will be useful for the task at hand. Some will not come from an appropriate trusted authority. Some will be redundant. Some will be irrelevant. Thus, before it is asserted into the inference engine's knowledge store, the semantic markup gleaned from each page must be filtered. The result will be a collection of facts and rules, which are likely to further the inferences being pursued, or serve as valuable relevance feedback terms. These facts and rules are passed to the inference engine, which may then iterate the entire process.

2.2 Using Text

The process described in the previous subsection makes no use of text, except to the extent that the result of markup swangling is a set of text terms. However, there is no reason that we cannot include appropriate text in the Web query. Adding text will influence the ordering of search results, possibly biasing them toward pages that will be most useful for the task at hand. Figure 2 shows how text can be included in the framework. First, a text query can be sent directly to the search engine (augmented by swangled markup, if such is available). Second, the extractor can pull text as well as markup out of retrieved pages. As with semantic markup, extracted text may be filtered or transduced in various ways before being used. Potentially useful filters include translation, summarization, trust verification, *etc.*

Incorporation of extracted text into the query of a subsequent round of processing corresponds to blind relevance feedback. The framework therefore provides a way to include both text and semantic markup as relevance feedback terms, even when the original query is homogeneous.

3. Three prototype systems

We have explored the problems and approaches to solving them through three prototype systems. While these systems do not exhaust the space of possibilities, they have challenged us to refine the techniques and provided valuable experience.

The first prototype, *OWLIR*, is an example of a system that takes ordinary text documents as input, annotates them with semantic web markup, swangles the results and indexes them in a custom information retrieval system. OWLIR can then be queried via a custom query interface that accepts free text as well as structured attributes.

Swangler, our second prototype, is a system that annotates RDF documents encoded in XML with additional RDF statements attaching swangle terms that are indexi-

ble by Google and other standard Internet search engines. These documents, when available on the web, are discovered and indexed by search engines and can be retrieved using queries containing text, bits of XML and swangle terms.

Our third prototype is Swoogle, a crawler-based indexing and retrieval system for RDF documents. It discovers RDF documents and adds metadata about them to its database. It also inserts them into a special version of the HAIRCUT information retrieval engine [21] that uses character n-grams as indexing terms.

3.1 OWLIR

OWLIR [23] is an implemented system for retrieval of documents that contain both free text and semantic markup in RDF, DAML+OIL or OWL. OWLIR was designed to work with almost any local information retrieval system and has been demonstrated working with two–HAIRCUT [21] and WONDIR. In this section we briefly describe the OWLIR system; readers are referred to Shah [23] for additional details.

While we have used OWLIR to explore the general issues of hybrid information retrieval, the implemented system was built to solve a particular task – filtering University student event announcements. Twice a week, UMBC students receive an email message listing 40-50 events that may be of interest, *e.g.*, public lectures, club meetings, sporting matches, movie screenings, outing, *etc.* Our goal is to automatically process these messages and produce sets of event descriptions containing both text and markup. These descriptions are then further processed, enriched with the results of local knowledge and inferencing and prepared for indexing by an information retrieval system. A simple form-based query system allows a student to enter a query that includes both structured information (*e.g.*, event dates, types, *etc.*) and free text. The form generates a query document in the form of text annotated with DAML+OIL markup. Queries and event descriptions are processed by reducing the markup to triples, enriching the structured knowledge using a local knowledge base and inferencing, and swangling the triples to produce acceptable indexing terms. The result is a text-like query that can be used to retrieve a ranked list of events that match the query.

OWLIR defines ontologies, encoded in DAML+OIL, allowing users to specify their interests in different events. These ontologies are also used to annotate the event announcements. Figure 3 shows a portion of the OWLIR Event Ontology, which is an extension to the ontologies used in ITTalks [8]. Events may be academic or non-academic, free or paid, open or by invitation. An event announcement made within the campus is identified as an instance of one of the natural kind of events or subcategories. Instances of subcategories are inferred to be a subtype of one of the natural kind of events.

Text Extraction. Event announcements are currently in free text. We prefer that these documents contain semantic markup. We take advantage of the AeroText™ system to extract key phrases and elements from free text documents. Document structure analysis supports exploitation of tables, lists, and other elements to provide more effective analysis.

We use a domain user customization tool to fine-tune extraction performance. The extracted phrases and elements play a vital role in identifying event types and adding semantic markup. AeroText has a Java API that provides access to an internal form of the extraction results. We have built DAML generation components that access this internal form, and then translate the extraction results into a corresponding RDF triple model that uses DAML+OIL syntax. This is accomplished by binding the Event ontology directly to the linguistic knowledge base used during extraction.

Inference System. OWLIR uses the metadata information added during text extraction to infer additional semantic relations. These relations are used to decide the scope of the search and to provide more relevant responses. OWLIR bases its reasoning functionality on the use of DAMLJessKB [17]. DAMLJessKB facilitates reading and interpreting DAML+OIL files, and allowing

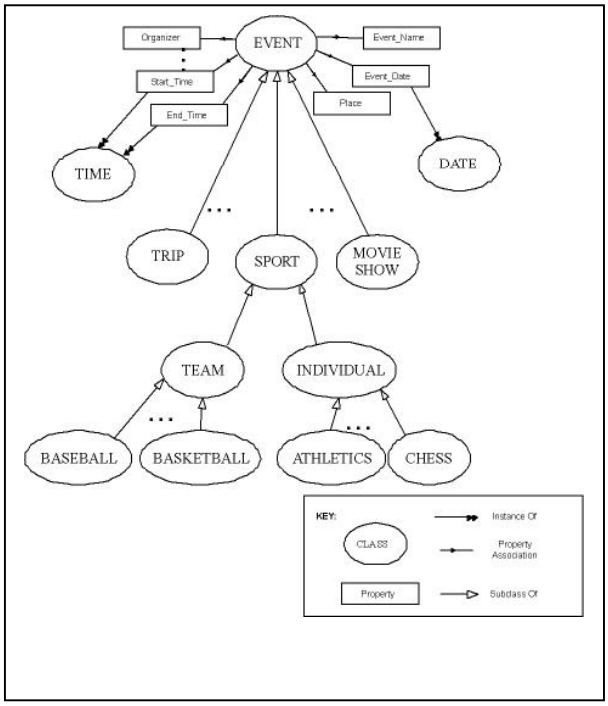


Figure 3. OWLIR annotations use terms from a DAML+OIL ontology of classes and properties that are useful in describing campus events.

the user to reason over that information. The software uses the SiRPAC RDF API to read each DAML+OIL file as a collection of RDF triples and Jess (Java Expert System Shell) [14] as a forward chaining production system to apply rules to those triples.

DAMLJessKB provides basic facts and rules that facilitate drawing inferences on relationships such as Subclasses and Subproperties. We enhance the existing DAMLJessKB inference capabilities by applying domain specific rules to relevant facts. For example, DAMLJessKB does not import facts from the ontology that is used to create instances; this limits its capacity to draw inferences. We have addressed this issue by importing the base Event ontology and providing relevant rules for reasoning over instances and concepts of the ontology. This combination of DAMLJessKB and domain specific rules has provided us with an effective inference engine.

As an example of the swangling process used in OWLIR, consider the markup, expressed here in RDF N3 notation, describing a movie with the title "Spiderman":

_j:00255 a owlir:movie; dc:title "Spiderman".

OWLIR has domain-specific rules that are used to add information useful in describing an event. One rule is triggered by a description of a movie event where we know the movie title. This rule requests that the Internet Movie Database (IMDB) agent seek additional attributes of this move, such as its genre. The results are added as triples, such as the following one (also in N3).

_:j00255 owlir:moviegenre "action".

This triple is then expanded with wildcards to generate seven terms, which are added to the document prior to indexing:

j00255.owlir.umbc.edu/event/moviegenre.action
*.owlir.umbc.edu/event/moviegenre.action
j00255.*.action
j00255.owlir.umbc.edu/event/moviegenre.*
j00255.*.*
.owlir.umbc.edu/event/moviegenre.
**.action

We conducted experiments with OWLIR to see if semantic markup within documents could be exploited to improve retrieval performance. We measured precision and recall for retrieval over three different types of document: text only; text with semantic markup; and text with semantic markup that has been augmented by inference. We used two types of inference to augment document markup: reasoning over ontology instances (*e.g.*, deriving the date and location of a basketball game); and reasoning over the ontology hierarchy (*e.g.*, a basketball game is a type of sporting event). For example, extracting the name of a movie from its description allows details about the movie to be retrieved from the Internet Movie Database site. A query looking for movies of the type *Romantic Genre* can thus be satisfied even when the initial event description was not adequate for the purpose.

We generated twelve hybrid (text plus markup) queries, and ran them over a collection of 1540 DAML+OIL-enhanced event announcements.

Unstructured data (e.g., free text)	Structured data with inferred data	Structured data plus free text
25.9%	66.2%	85.5%

Table 1. Mean average precision over twelve hybrid queries given to OWLIR.

Indexed documents contain RDF Triples and RDF Triple Wildcards. This gives users the flexibility to represent queries with RDF Triple wildcards. DAML+OIL captures semantic relationships between terms and hence offers a better match for queries with correlated terms.

These experiments were run using the WONDIR information retrieval engine. Preliminary results are shown in Table 1 and in Shah *et al.* [23]. Retrieval times for free text documents and documents incorporating text and markup are comparable. Including semantic markup in the representation of an indexed document increases information retrieval effectiveness. Additional performance benefits accrue when inference is performed over a document's semantic markup prior to indexing. While the low number of queries at our disposal limits any conclusions we might draw about the statistical significance of these results, we are nonetheless strongly encouraged by them. They suggest that developing retrieval techniques that draw on semantic associations between terms will enable intelligent information services, personalized Web sites, and semantically empowered search engines.

3.2 Swangler

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.

Some Internet search engines, such as Google, do in fact discover and index RDF documents. There are several problems with the current situation that stem from

the fact that systems like Google treat semantic web documents (SWDs) as simple text files. One simple problem is that the XML namespace mechanism is opaque to these engines. A second problem is that the tokenization rules are designed for natural languages and do not always work well with XML documents. Finally, we would like to take advantage of the semantic nature of the markup.

We have applied the swangling technique to SWDs to enrich them with additional RDF statements that add swangle terms as additional properties of the documents. As with OWLIR, each swangle term encodes one triple or a triple with one or more of its components replaced with a special don't care URI (rdf:Resource, in this case). For example, the RDF triple

http://www.xfront.com/owl/ontologies/camera/#Digital
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://www.xfront.com/owl/ontologies/camera/#PurchaseableItem

is used to generate the seven possible combinations of the subject, predicate and object with a don't care URL (the triple with all don't care URLs is not used). The concatenation of the URLs in each triple is then hashed and converted to a base-32 number. This example results in the seven swangle terms as follows:

BE52HVKU5GD5DHRA7JYEKRBFVQ
WS4KYRWMO3OR3A6TUAR7IIIDWA
2THFC7GHXLRMISEOZV4VEM7XEQ
HO2H3FOPAEM53AQIZ6YVPFQ2XI
6P3WFGOWYL2DJZFTSY4NYUTI7I
N656WNTZ36KQ5PX6RFUGVKQ63A
IIVQRXOAYRH6GGRZDFXKEEB4PY

A simple ontology[2] is used to provide an RDF vocabulary for annotating the original document with the generated swangle terms.

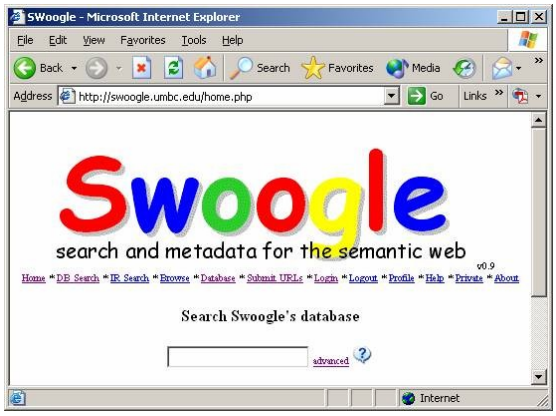
The RDF files are modified to include the additional statements and left on the web for the Google spider to discover. When discovered, Google indexes the contents including the swangle terms. These can be subsequently used to retrieve the documents through a simple interface that takes user provided triples, swangles them, and composes a query using the resulting terms.

A Java application was developed that implements swangling. It allows for the swangling of an RDF-based semantic web document and outputting the annotated, swangled document. The source code and documentation for this application are available at the *Semantic Web Central* web site (http://semwebcentral.org/).

[2] http://swoogle.umbc.edu/ontologies/swangle.owl

3.3 Swoogle

Since the current semantic web consists of documents encoded in RDF, it is worth considering what a specialized indexing and retrieval engine for these semantic web documents (SWDs) might be like. Search engines for SWDs could exploit the fact that the documents they encounter are designed for machine processing and understanding. Conventional search engines can not do much to interpret the meaning of documents because the state of the art in natural language processing is not up to the task. Even if it were, the computational cost for interpreting billions of documents would be prohibitive in any foreseeable future. SWDs, on the other hand, are encoded in languages designed for machine interpretation and understanding. While full processing of their content is still a challenging and expensive task, the barriers are significantly lower. In particular, it is relatively easier to discover and compute interesting and useful metadata about the SWDs, such as their intended use (e.g., as an ontology, as instance data or as a mapping between two ontologies).



Swoogle is a crawler based search engine for RDF documents available at http://swoogle.umbc.edu/.

We have built Swoogle[3] [12] as a prototype internet indexing and retrieval engine for semantic web documents encoded in RDF and OWL. The system is intended to support human users as well as software agents and services. Human users are expected to be semantic web researchers and developers who are interested in accessing, exploring and querying a collection of metadata for a collection of RDF documents automatically discovered 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.

[3]The Swoogle semantic web indexing and retrieval system can be accessed at http://swoogle.umbc.edu/

The system consists of a database that stores metadata about the SWDs, several 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-based and web service APIs to provide useful services. A key metadata property we compute of a SWD is its "rank". Like the Page Rank [5a] concept, our SWD rank is a measure of the semantic web document's "importance" or "popularity". We have used this measure to order results returned by the retrieval engine. This algorithm takes advantage of the fact that the graph formed by SWDs has a richer set relations that that formed by a collection of simple hypertext documents. 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 property).

We envision the following several broad uses of a retrieval system like Swoogle: finding appropriate ontologies, finding instance data and studying the structure of the semantic web.

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. This has contributed to the proliferation of ontologies, since developers ignorant of the extant ontologies just write their own. 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.

The semantic web seeks to enable the integration of distributed information. But 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.

The metadata computed by Swoogle will provide structural information about the semantic web, such as 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?

4. Discussion

Our experience in building and evaluating these systems has helped us to understand some of the dimensions inherent in adapting information retrieval to the semantic web. We will briefly describe them as well as some of the related issues and decisions that arise.

The first dimension involves what kind of documents we expect, i.e., RDF documents encoded in XML (or perhaps N3 or some other standard encoding) or text documents with embedded RDF markup. Swoogle and Swangler are designed to work only on well formed RDF documents whereas OWLIR can handle compound documents with both text and RDF intermixed.

The second dimension concerns how the semantic web markup is processed – as structured information with an underlying data/knowledge model or as text with little or no associated model. OWLIR and Swangler treat markup as structured information and perform inferences over it following the semantics of RDF and OWL. The resulting data is ultimately reduced to swangle terms which, while a lossy transformation, still preserves much of the information. Swoogle has components on both ends of this spectrum. It stores metadata about RDF documents in its database in a way completely faithful to its structure and meaning. This allows it to retrieve documents based on the set of classes, properties and individuals mentioned in them or implied by the semantic model. In this way, Swoogle treats an RDF documents as a "bag of URIs" just as a conventional IR systems treats a text document as a "bag or words". Swoogle also treats RDF documents (in their canonical XML encoding) as text documents which are indexed by the HAIRCUT retrieval engine.

The final dimension delineates systems using conventional retrieval components and infrastructure from those that use specialized IR systems to handle semantic web documents. Swangler was designed with goal of enabling Google and other Internet search engines to index semantic web documents. OWLIR and Swoogle, on the other hand, use special retrieval engines adapted to handle the task of indexing and retrieving documents with RDF markup.

In the remainder of this section, we will introduce and discuss some additional issues that have surfaced in our work.

4.1 Tokenization

Most search engines are designed to use words as tokens. There are two immediate issues that present themselves when considering the conversion of RDF triples into swangle terms that look like indexing terms to a Web search engine – which triples should be selected for swangling and what techniques should be used to swangle a selected triple.

What to swangle. Some search engines, such as Google, limit query size. Care must be taken to choose a set of triples that will be effective in finding relevant documents. Some triples carry more information that others. For example, every instance is a type of owl:thing, so adding triples asserting owl:thingness will not be very helpful, especially if the query size is limited. OWL and RDF descriptions typically contain anonymous nodes (also know as "blank nodes") that represent existentially asserted entities. Triples that refer to blank nodes should probably be processed in a special way, since including the "gensym" tag that represents the blank node carries no information. It might be possible to develop a statistical model for OWL annotations on documents similar to statistical language models. Such a model could help to select triples to include in a query.

How to swangle. In the OWLIR system we explored one approach to swangling triples. More experimentation is clearly needed to find the most effective and efficient techniques for reducing a set of triples to a set of tokens that a given information retrieval system will accept. The simplest approach would be to decompose each triple into its three components and to swangle these separately. This loses much of the information, of course. OWLIR followed an approach which preserved more information. Each triple was transformed into seven patterns, formed by replacing zero, one or two of its components with a special "don't care" token. Each of the seven resulting tokens was then reduced to a single word-like token for indexing.

4.2 Reasoning and trust

When to reason. We have a choice about when to reason over Semantic Web markup. We can reason over the markup in a document about to be indexed, resulting in a larger set of triples. We can also reason over a query that contains RDF triples prior to processing it and submitting it to the retrieval system. Finally, we can reason over the markup found in the documents retrieved. In OWLIR, we chose to reason both over documents as they were being indexed and over queries about to be submitted. It is not obvious to us how much redundancy this entails nor is it clear if there is a best approach to when to do the reasoning.

How much to reason. A similar problem arises when one considers how much reasoning to do or whether to rely largely on forward chaining (as in OWLIR) or a mixture of forward and backward reasoning.

What knowledge to trust. The information found on the Semantic Web will vary greatly in its reliability and veracity, just as information on the current Web. It will not do just to inject into our reasoning the facts and knowledge from a newly found and relevant document. Moreover, we may need to take care not to create an inconsistent knowledge base. This problem is being studied in the context of models of trust on the Web [11][16].

Much of the information found in a document comes from somewhere else – typically another document. Data provenance [6] is a term used for modeling and reasoning about the ultimate source of a given fact in a database or

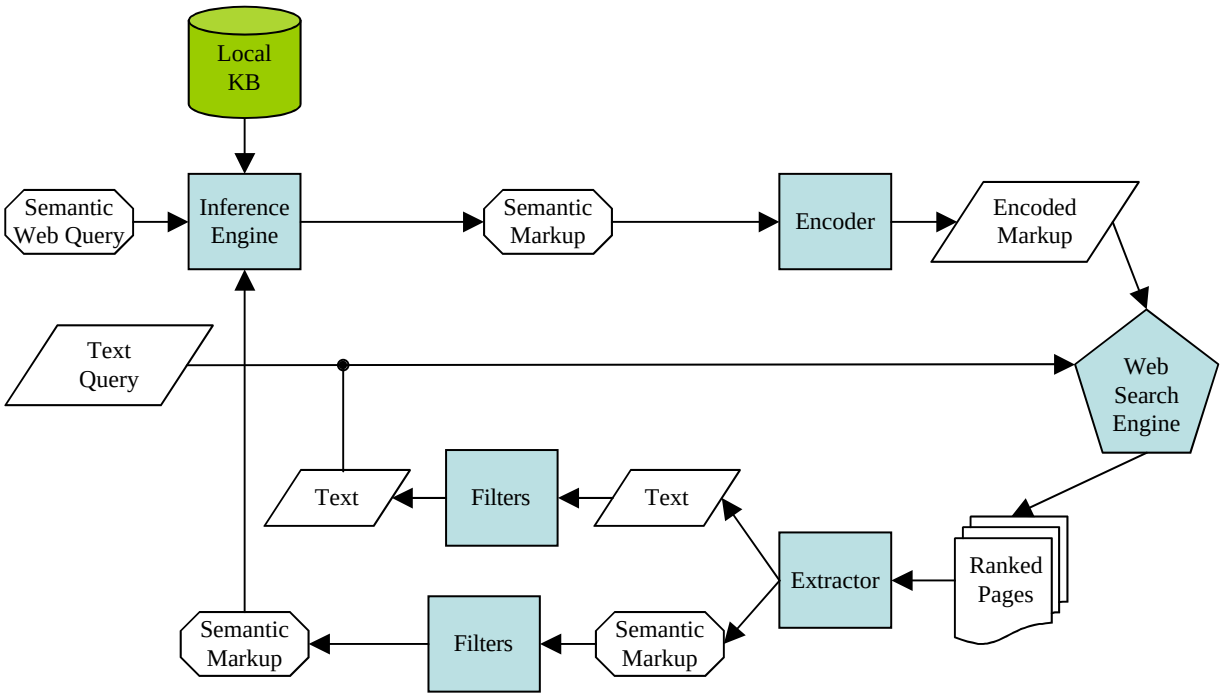


Figure 2. Text can also be extracted from the query results, filtered, and injected into the query.

document. For systems that extract and reason about facts and knowledge found on the Semantic Web, it will be important to (i) inform our trust model and make better decision about the trustworthiness of each fact; and (ii) remove duplicate facts from our semantic model.

4.3 Dealing with search engines

Control. The basic cycle we've described involves (re)forming a query, retrieving documents, processing some of them, and repeating. This leaves us with a decision about whether to look deeper into the ranked result set for more information to use in reforming our query, or to reform the query and generate a new result set. The choice is similar to that faced by an agent in a multiagent system that must decide whether to continue reasoning with the information it has or to ask other agents for more information or for help with the reasoning [20]. We need some metric that estimates the expected utility of processing the next document in the ranked result set.

Spiders. Web search engines typically do not process markup. So, we need a way to give a search engine spider a preprocessed (swangled) version of a Web page when it tries to spider it for indexing. This can be easily accomplished if we have control of the HTTP server that serves a page – it checks to see if the requesting agent is a spider. If so, it returns the swangled version of the page, otherwise it returns the original source page. The preprocessing can be done in advance or on demand with caching.

Offsite annotation. The technique described above depends on having control over all of the servers associated with a Semantic Web page. If this is not the case, some work arounds are needed. One option is to mirror the pages on a server that does automatic swangling. The pages should have a special annotation (*e.g.,* in RDF) that asserts the relationship between the source and mirrored pages.

Search engine limitations. Web based search engines have limitations that must be taken into account, including how they tokenize text and constraints on queries. We would like swangled terms to be accepted as indexable terms by typical search engines. The two retrieval systems we used in OWLIR were very flexible in what they accepted as a token; tokens could be of arbitrary length and could include almost any non-whitespace characters. Many commercial systems are much more constrained. With Google, for example, we were advised to keep the token length less than 50 and to include only lower and uppercase alphabetic characters. Many commercial systems also limit the size of a query to a maximum number of terms. Google, for example, currently has a limit of ten terms in a query. These limitations, as well as others, affect how we have to interface to a given retrieval engine.

5. Conclusion

The Semantic Web will contain two kinds of documents. Some will be conventional text documents enriched by annotations that provide metadata as well as machine interpretable statements capturing some of the meaning of the documents' content. Information retrieval over collections of these documents offers new challenges and new opportunities. We have presented a framework for integrating search and inference in this setting that supports both retrieval-driven and inference-driven processing, uses both text and markup as indexing terms, exploits today's text-based Web search engines, and tightly binds retrieval to inference. While many challenges must be resolved to bring this vision to fruition, the benefits of pursuing it are clear. The Semantic Web is also likely to contain documents whose content is entirely encoded in an RDF based markup language such as OWL. We can use the swangling technique to enrich these documents to terms that capture some of their meaning in a form that can be indexed by conventional search engines. Finally, there is also a role for specialized search engines that are designed to work over collections of RDF documents.

6. Acknowledgements

Partial research support provided by DARPA contract F30602-00-0591 and NSF award IIS-0326460. We acknowledge many contributions from colleagues in the UMBC ebiquity research group and in the Distributed Information Systems section of the Johns Hopkins University Applied Physics Laboratory.

7. References

[1] Abiteboul, S., Quass, D., McHugh, J. Widom, J. and Wiener, J. 'The Lorel query language for semistructured data.' *International Journal on Digital Libraries* 1, pages 68-88, April 1997.

[2] Arocena, G. and Mendelzon, A. 'WebOQL: Restructuring documents, databases and webs.' In *International Conference on Data Engineering,* pages 24-33. IEEE Computer Society, 1998.

[3] Bar-Yossef, Z., Kanza, Y., Kogan, Y., Nutt, W. and Sagiv, Y.. 'Quest: Querying semantically tagged documents on the World Wide Web.' In *Proc. of the 4th Workshop on Next Generation Information Technologies*

and Systems, volume NGITS'99, Zikhron-Yaakov (Israel), July 1999.

[4] Berners-Lee, T. and Fischetti, M. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor. Harper, San Francisco. 1999.

[5] Berners-Lee, T., Hendler, J. and Lassila, O. 'The Semantic Web.' *Scientific American*, May 2001.

[5a] Brin, Sergey and Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Proceedings of the 7th international conference on World Wide Web, Elsevier Science Publishers B. V. pp 107-117, Brisbane, Australia, 1998.

[6] Buneman, P., Khanna, S. and Tan, W-C. 'Why and Where: A Characterization of Data Provenance.' *International Conference on Database Theory (ICDT)* 2001.

[7] Chinenyanga, T. and Kushmerick, N. 'Elixir: An expressive and efficient language for XML information retrieval.' In *SIGIR Workshop on XML and Information Retrieval*, 2001.

[8] Cost, R. S., Finin, T., Joshi, A., Peng, Y., Nicholas, C., Soboroff, I., Chen, H., Kagal, L., Perich, F., Zou, Y., and Tolia, S. 'ITTALKS: A Case Study in the Semantic Web and DAML+OIL.' *IEEE Intelligent Systems* 17(1):40-47, 2002.

[9] Davies, J., Weeks, R. and Krohn, U. 'QuizRDF: Search technology for the Semantic Web.' In *WWW2002 Workshop on RDF and Semantic Web Applications*, Hawaii, 2002.

[10] Deutsch, A.,Fernandez, M., Florescu, D., Levy, A. and Suciu, D. 'XML-QL: A query language for XML.' In *Proceedings of the Eighth International World Wide Web Conference*, 1999.

[11] Ding, L,. Lina Zhou, and Tim Finin, 'Trust Based Knowledge Outsourcing for Semantic Web Agents,' *2003 IEEE/WIC International Conference on Web Intelligence (WI 2003)*, October 2003, Halifax, Canada.

[12] Ding, L., Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Joel Sachs, Vishal Doshi, Pavan Reddivari, and Yun Peng, Swoogle: A Search and Metadata Engine for the Semantic Web, Thirteenth ACM Conference on Information and Knowledge Management (CIKM'04), Washington DC, November 2004.

[13] Egnor, D. and Lord, R. 'Structured information retrieval using XML.' In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, Athens, Greece, July 2000.

[14] Friedman-Hill, E. *Jess, the Java expert system shell.* Sandia National Laboratories. 2000.

[15] Fuhr, N. and Grojohann, K. 'XIRQL: An extension of XQL for information retrieval.' In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, Athens, Greece, July 2000.

[16] Golbeck, J., Parsia, B., and Hendler, J. 'Trust networks on the Semantic Web.' To appear in *the Proceedings of Cooperative Intelligent Agents 2003*, August 27-29, Helsinki, Finland.

[17] Kopena, J. and Regli, W., 'DAMLJessKB: A tool for reasoning with the Semantic Web.' *IEEE Intelligent Systems* 18(3), May/June, 2003.

[18] Kwok, C., Etzioni, O. and Weld, D. 'Scaling question answering to the Web. ' In *Proceedings of WWW10*, Hong Kong, 2001.

[19] Mayfield, J. 'Ontologies and text retrieval.' *Knowledge Engineering Review* 17(1):71-75. 2002.

[20] Mayfield, J., Finin, T., Narayanaswamy, R., Shah, C., MacCartney, W. and Goolsbey, K. 'The Cycic Friends Network: Getting Cyc agents to reason together.' *Proceedings of the CIKM Workshop on Intelligent Information Agents*. 1995.

[21] Mayfield, J., McNamee, P. and Piatko, C. 'The JHU/APL HAIRCUT system at TREC-8.' *The Eighth Text Retrieval Conference (TREC-8)*, pages 445-452, November 1999.

[22] Mayfield, J. and Tim Finin, Information retrieval on the Semantic Web: Integrating inference and retrieval, SIGIR Workshop on the Semantic Web, Toronto, 1 August 2004

[23] Reagle, J. (ed.), *RDF in XHTML*. W3C Task Force Document, May 2003.

[24] Shah, U., Finin, T., Joshi, A., Cost, R. S. and Mayfield, J. 'Information Retrieval on the Semantic Web.' *10th International Conference on Information and Knowledge Management*, November 2002.