SWHi System Description: A Case Study in Information Retrieval, Inference, and Visualization in the Semantic Web

Ismail Fahmi, Junte Zhang, Henk Ellermann, and Gosse Bouma

Information Science Department and University Library,
University of Groningen
Broerstraat 4, 9712 CP Groningen, The Netherlands
{i.fahmi,junte.zhang,h.h.ellermann,g.bouma}@rug.nl
http://www.rug.nl

Abstract. Search engines have become the most popular tools for finding information on the Internet. A real-world Semantic Web application can benefit from this by combining its features with some features from search engines. In this paper, we describe methods for indexing and searching a populated ontology by using an information retrieval tool; its results are enriched with inference. For visualization purposes, all of the retrieved ontology instances are clustered based on their classes; and the clusters are linked using instance properties. The approach is illustrated using our SWHi (Semantic Web for History) prototype as a case study.

Keywords: semantic web, ontology, information retrieval, inference, visualization.

1 Introduction

The Semantic Web can be seen as a general web which describes units of information. Once this information is available in some web documents (which then become Semantic Web documents), people can gather and manipulate the exchanged information using Semantic Web technologies in various useful ways. Tim Berners-Lee in his Scientific American article "the Semantic Web" [2] illustrated some examples of how the semantic information can help everyday tasks, such as finding a health care provider, prescription treatments, making an appointment, and planning a trip.

In order for Semantic Web technologies to have an impact, they should be able to work together with existing general information retrieval technologies [6], and even provide new services which cannot be delivered by general Web search engines such as Google. The most popular Web search engine, Google[1] combines instant responses, huge repositories, advanced search methods, and a sophisticated relevance-ranking algorithm.

[1] Google www.google.com

We begin the development of our Semantic Web application from a library point of view. In this case, Eric Miller [9] believes that "Libraries–digital libraries in particular–are important memory organizations that form a keystone for the development of the Semantic Web." The digital library stands on collections of annotated data, in the form of metadata. This should naturally make the digital library a successful primary adopter of the Semantic Web, and, on the other hand, challenge the Semantic Web to improve and widen services provided by digital libraries, which nowadays still heavily rely on information retrieval technology.

In our Semantic Web for History (SWHi) project, we combine some features from the Web search engine and the Semantic Web technology. From the Semantic Web technology point of view, the adoption of the Web search engine technology is expected to improve its search performance. In this paper we describe how the search engine tool Lucene[2] can be used to index ontology instances, parse user input queries, and retrieve matched instances. Given a plain list of ontology instances from the search results, Semantic Web technology will enrich the retrieved information. Inference will be applied during the indexing and enrichment steps. For visualization purposes, we organize the results in clusters based on classes of the retrieved instances (e.g. person, organization, document, subject, and year) and relations between instances in the classes. We present the clusters and their relationships using a two dimensional cluster map. Through this map, users can browse search results interactively, and explore interesting relationships in an ontology.

2 Motivation

2.1 Data Sources

The concept of a Semantic Web is promising but difficult to implement on most current Web documents. The Semantic Web requires semantic information which is typically not encoded in the documents. Considering the importance of such information, metadata is added into the document. On the other hand, many digital libraries do have metadata in place. Metadata is a key piece which describes every resource managed by the digital libraries.

Our SWHi application is developed from the digital library point of view, where our main data sources are repositories which provide metadata (subsection 3.2). This metadata is mapped and stored into an ontology based on an ontology schema. Furthermore, literal values in the metadata, for example describing title and description properties, are analyzed, from which we extract named entities, events and terminology. To enrich the ontology, we also extract new related information from selected Web documents.

2.2 Information Retrieval in the Semantic Web

Current popular web search engines (e.g. Google and Yahoo[3]) provide both simple and advanced search interfaces. While the advanced search interfaces

[2] Lucene lucene.apache.org

[3] Yahoo www.yahoo.com

provide more functionality, the simple search interfaces are preferred by most users. A Semantic Web application will typically also provide these kinds of search interfaces.

Using a simple search interface, a user can enter query terms regardless of the question in which fields the terms would exist. For example, a user may want to find any information (any document, year, person, or relation between persons) in our SWHi ontology related to a topic such as *French settling colonies involving Mr. Samuel Kirkland and General Washington in 1777.* Using a bag-of-words searching technique, she might simply type *kirkland washington 1777 french settle* into a search form. We face some issues while processing this query using an RDF query language such as SeRQL[3]. A query processor (which generates SeRQL queries) will face at least two problems. First, it does not know in which class or property a word can be found. To avoid this problem, a Semantic Web application such as OpenAcademia[4] requires users to type a keyword into the appropriate field (*author, title* or *year*) in its advanced search interface. Second, there are some limitations in the substring matching of SeRQL using a wildcard character '*'. Searching for *general** will match *general* and *generally* only at the beginning of a text. And searching for ** general ** (with a space between the wildcards and the word) will only match *general* in the middle of a text, but not at the beginning or end of the text. These problems can be solved using an information retrieval application such as Lucene which provides powerful, accurate, and efficient search algorithms. Besides the fielded searching feature, Lucene also supports phrase queries, wildcard queries, proximity queries, range queries and more[5].

Prior uses of information retrieval technology in a Semantic Web application can be found in QuizRDF[4], the Knowledge and Information Management (KIM) platform[1], OWLIR and Swoogle[6]. QuizRDF creates RDF resource indexes based on RDF Schema and retains this structure in its indexes. KIM uses Lucene engine to index and retrieve semantically annotated documents while OWLIR and Swoogle use the Haircut information retrieval engine to index and retrieve RDF documents based on character n-grams as indexing terms.

2.3 Visualization

In the Semantic Web, visualization is becoming more important. In our case, since the retrieved instances can be of any type (documents, persons, years, etc.), a common plain list presentation is not suitable. There are complex relationships among the resource instances which cannot be presented using a plain list. Moreover, this presentation typically only displays a small number of search results (in the range of 10-20 results per page). Documents obscured in the tail of a search result will likely never be accessed.

Various solutions to this problem have been proposed in the IR as well as the SW area. Currently, popular result presentations in the information re-

[4] OpenAcademia www.openacademia.org
[5] Lucene's Features lucene.apache.org/java/docs/features.html

trieval technology use topical clustering and mapping techniques. Vivisimo[6] and Grokker[7], for instance, both use clustering techniques to analyze and organize search results according to topics found in the retrieved document descriptions.

In the Semantic Web, the complex nature of relationships between concepts in an ontology has driven many efforts toward graphical visualization of ontology browsing and navigation [10,12,11,7]. For example, Cluster Map [7] is used to visualize instances of selected classes, organized by their classifications. This map is designed to aid users when navigating their search results and ontologies. The Spring embedding model [8,5] has been widely used to visualize collections of instances and ontologies[10,7]. It draws highly related entities close to each other with a directed edge and gives the effect of separation in a two-dimensional plane.

3 System Architecture and Data Source

3.1 Architecture

The general architecture of the SWHi system is shown in Fig. 1. It consists of three layers: Knowledge Management System (KMS), Semantic Web Application, and User Interface layers. The bottom layer, is responsible for processing (information extraction and semantic annotation), indexing, and storing historical resources (metadata and documents), also providing API for its upper layer to query the knowledge base. This layer highly depends on several third party tools, such as GATE[8], Sesame[9], and Lucene.

In the middle layer, several application modules which enable the Semantic Web were developed to carry out the following functions: processing data sources (text and metadata), processing user queries, and delivering results in several ways (network graph, cluster map, and time line). And the top layer provides interface to users which are being designed as simple and easy to use as possible.

3.2 Data Source and Ontology

It is obvious from the Fig. 1 that the ontology plays as a central role in the SWHi system. For the development of the SWHi ontology, we reuse existing ontology resources for structuring and storing historical information, namely: PROTON[10] base ontology, the types of American history imprints identified (automatically) in the metadata, the taxonomical subject classification by NewsBank/Readex[11], Dublin Core[12], and Friend of a Friend[13]. This ontology is stored using the Sesame 2, an RDF storage and querying framework.

[6] Vivisimo www.vivisimo.com
[7] Grokker www.grokker.com
[8] GATE gate.ac.uk
[9] Sesame www.openrdf.org
[10] PROTo ONtology proton.semanticweb.org
[11] Newsbank InfoWeb infoweb.newsbank.com/?db=EVAN
[12] Dublin Core dublincore.org
[13] FOAF xmlns.com/foaf/0.1/

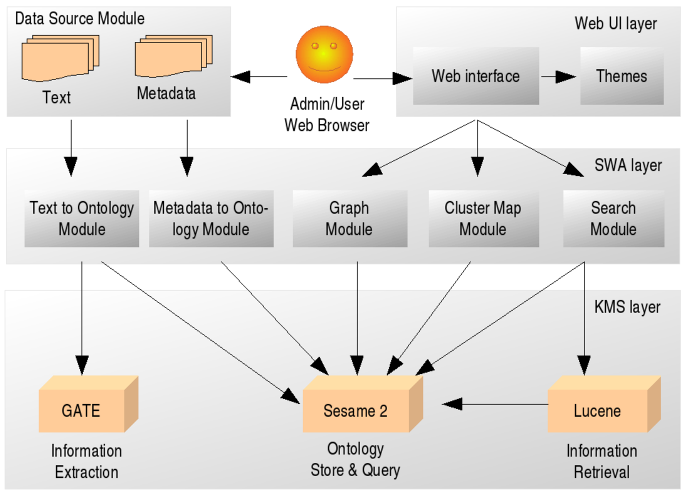


Fig. 1. The general architecture of the SWHi system

Our initial instances for this ontology are extracted from the Early American Imprints, Series I: Evans, 1639-1800[14]. This knowledge source gives insight in all published works of the 17th- and 18th-century America. Its metadata consist of 36,305 records, which are elaborately described (title, author, publication date, etc) with numerous values, and have been compiled by librarians in the format MARC21. In the future we will extend the data sources with other historical electronic journal metadata as well as full texts, like digitized version of Early American Imprints, Wikipedia, and biographical profiles that libraries have about historical US figures.

4 Indexing, Searching, and Inference

4.1 Indexing and Inference

Performance is very important in a real-word application. No matter how sophisticated and accurate the logic behind a search engine, if it cannot provide a fast search response, users will likely less appreciate it. To ensure that our system will have an acceptable response time and process query terms efficiently, we use the Lucene text search engine API to index our ontology instances. We make use of Lucene fielded data feature to store and index instance properties. When indexing, each instance in the SWHi ontology is added to a Lucene index as if it is a new document, and instance properties are added to the document as document fields.

In our experiments, getting information directly from an ontology through inference could cost an unacceptable processing time, especially if the inference

[14] Early American Imprints Series I www.readex.com

queries are complex or repeated many times. However inference is required to return the most relevant instances given user queries. For example, searching for instances with literal values containing the term “washington” in the SWHi ontology would give us 7 instances of Person class or 586 instances of all classes (Person, Location, Event, and Document) in any order. Using inference, instances with higher relevance can have higher position in the order. For example, a person who is known by many people and created many documents would get a higher score. For this purpose, we apply inference during an indexing step to get additional information about an instance.

Table 1 illustrates how an instance of Person class should be indexed by Lucene. For example, an instance with a label “George Washington” is being indexed. Each Lucene field of the instance is populated by querying the ontology repository. Then, the number of persons that Washington knows will be stored in the `foaf_knows` field, and the number of persons who know him will be stored in the `known_by` field. These numbers will not be searched, but will be used during query analysis which boost particular fields based on their values.

Table 1. The Lucene fields of Person class instance. The `container` field will used by search “All”

Field	**Lucene type**	**Description**
uri	Field.UnStored	a URI of an instance
rdfs_label	Field.Text	a short label of an instance
container	Field.Text	contains all data from other fields (not stored)
foaf_topic_interest	Field.Text	a textual list of topics
foaf_knows	Field.Keyword	the number of persons as String
known_by	Field.Keyword	the number of persons as String
dc_creator	Field.Text	a textual list of document titles
protont_involvedIn	Field.Text	a textual list of events
protont_startTime	Field.Keyword	a date as a String (YYYYMMDD)
protont_endTime	Field.Keyword	a date as a String (YYYYMMDD)

4.2 Searching and Enrichment

Retrieving and processing information from the SWHi ontology will be done through these processes: Query formulation, Query analysis and parsing, Search and retrieval, Enrichment, and Clustering.

Query Formulation. When performing a query, users often encounter difficulties whether they have to use “AND” or “OR” [13]. To overcome this problem, we present a *free-text search* interface to users. Using a form in this interface, users can type any query terms, such as a combination of terminology, person name, year, and location. For advanced users, we provide an advanced search interface where they can use multiple fields for their query terms.

Query Analysis and Parsing. All free-text search queries will be processed by Lucene. Since its searches are case-sensitive, a general best practice is to lowercase query terms during query analysis. The StandardAnalyzer which is used during the indexing will be used again to perform this task. In this step, we set different boosting factors to the index fields based on their importance to the class being searched. For example, the field `known_by` is a good indicator of how well-known the person was. This can be implemented using `FunctionQuery` feature of Lucene which can return a score based on fields' values. For advanced search queries, we generate SPARQL queries and send them directly to the Sesame repository.

Search and Retrieval. Given a free-text search query, Lucene searches its index to find all matched resources, and given an advanced search query, Sesame searches for instances from its ontology repository. Each time a search is performed, the Search Module retrieves URIs of instances in the search results and stores them into a cache memory. This will speed up the retrieval process when a user clicks on other pages of the same search results, which could happen if the number of instances exceed an allowed number of instances per page.

Enrichment. The goal of this step is to deduce new information given a list of URIs in the search results retrieved by the previous search process. The SWHi ontology will be used to enrich presented instances with important information and relationships. For this purpose, we define inference algorithms for each instance class. Since the search phase typically produces many search results, we have to optimize this enrichment phase to achieve an acceptable performance. For example, we limit only to the first 200 instances returned by Lucene that will be enriched for clustering, and 10 to 20 instances for a plain list presentation.

Clustering and Visualization. Clustering is performed to preprocess the results before they are presented in a cluster map to users. Since the results consist of various resource types, a visual representation technique would be a promising option. The underlying concept of our visualization is based on this Visual Information Seeking Mantra [13]: "*Overview first, then zoom & filter, then details on demand*".

We implement this principle by clustering the results. Our cluster data will be organized to support the following levels of presentation details:

Global view of results. Users can see a global view of all of the results in a two-dimensional graphical visualization. In this view, the results are organized into clusters based on the results' classes or topics. Grouping by topics seems to be more interesting than by classes, because users can see interactions between instances of different classes in a topic. Between the clusters, we draw relationships based on properties carried by the results in each cluster. Every instance in a cluster will be presented using a symbol.

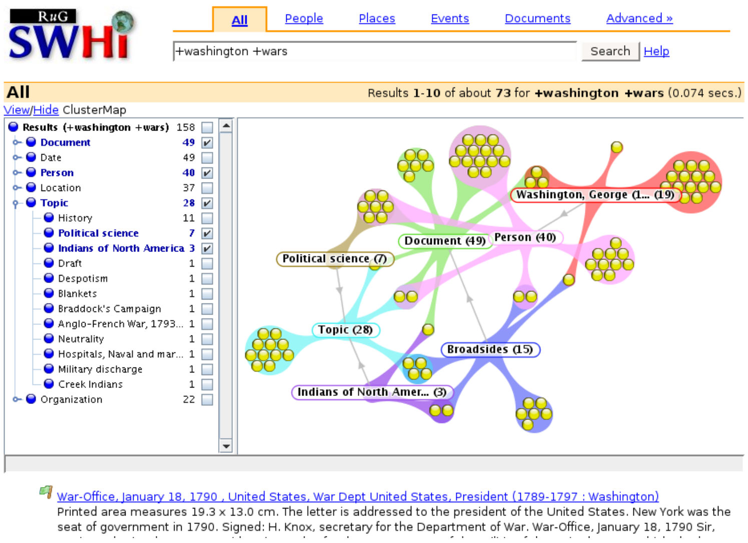
Zoom view of a cluster. Users can zoom into a cluster to see interactions between instances in the selected cluster. For example, in a cluster of persons, users can see how persons in that cluster know each other.

Detailed description view of a cluster or a node. Users can read a detailed description of a cluster or a node (member of a cluster). For example, they can see the name of a cluster, the number of its nodes (cluster's members), a complete list of the nodes' titles and links, or a short description of a node.

5 Results

In this section, we describe the results of our system development and illustrate them with some examples. Assume users want to find information about "George Washington and wars" and type the keywords "+washington +wars" in a free-text search interface. After receiving search results, the first task of the user interface is to display a general view of the results. For this task we use the Cluster Map [7] as shown in Fig. 2.

The user interface in the figure is divided into three main panes: left, right, and bottom. The left pane shows the classification tree of the results, containing the names of the clusters, their children and the numbers of objects within each cluster. In this pane, the users perform cluster selection that will change the visualization of the clusters in the right pane. For example, *Document*, *Person*, and *Topic* clusters are selected together with some of their sub-clusters as shown in the right pane. Objects in the results are now classified according to their



All Results **1-10** of about **73** for **+washington +wars** (0.074 secs.)

Results (+washington +wars)	158	☐
Document	**49**	✓
Date	49	☐
Person	**40**	✓
Location	37	☐
Topic	**28**	✓
History	11	☐
Political science	**7**	✓
Indians of North America	**3**	✓
Draft	1	☐
Despotism	1	☐
Blankets	1	☐
Braddock's Campaign	1	☐
Anglo-French War, 1793...	1	☐
Neutrality	1	☐
Hospitals, Naval and mar...	1	☐
Military discharge	1	☐
Creek Indians	1	☐
Organization	22	☐

War-Office, January 18, 1790 , United States, War Dept United States, President (1789-1797 : Washington)
Printed area measures 19.3 x 13.0 cm. The letter is addressed to the president of the United States. New York was the seat of government in 1790. Signed: H. Knox, secretary for the Department of War. War-Office, January 18, 1790 Sir,

Fig. 2. Search results for the query "+washington +wars" visualized using ClusterMap

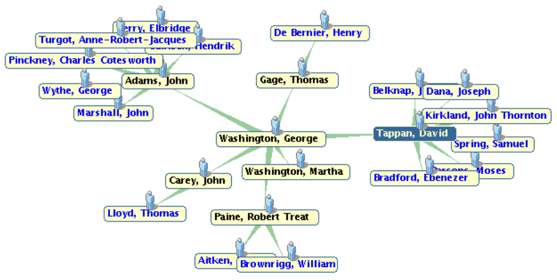


Fig. 3. Browsing objects and their relationships

clusters and intersections, which help users select objects satisfying their needs. In the figure they can click on a cluster of 7 objects (documents) related to a *Political science* topic. A list of objects (of any type, e.g., document, person, date, or location) in the cluster are then displayed in the bottom pane. Detailed information of an object will be presented when users click on a title in the list.

A different visualization strategy is shown in Fig. 3. While ClusterMap shows objects in the result set as clusters, the last strategy shows relationships between objects in a cluster or in the whole result set. For example, the figure shows how individuals in the *Person* class relate each other. Objects and their relations are not limited by the result set, but can be retrieved directly from the repository when requested information is not available in the set. This can be seen as another way of browsing the result set and repository. We implement this strategy using the TouchGraph[15] tool.

6 Summary and Future Work

This paper has described a case study in implementing information retrieval, inference, and visualization in the Semantic Web. The use of information retrieval technology is mainly motivated by pragmatic reasons which are to provide rich search functionalities and to return semantically related search results with high performance. However, in this application, the ontology keeps playing an important role, especially in shaping information structure in the indexes, in inferencing, and in clustering. Subject and terminology classes in the ontology help generating clusters based on resource topics. Visualization based on this grouping technique is interesting because it combines information from the ontology and information retrieval into a single graph.

The SWHi application is an on going project. We are also in a progress of enriching the ontology by implementing named entity, event, and terminology extraction.

[15] TouchGraph www.touchgraph.com

References

1. Kiryakov Atanas, Popov Borislav, Terziev Ivan, Manov Dimitar, and Ognyanoff Damyan. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2, 2005.
2. Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 2001.
3. Jeen Broekstra and Arjohn Kampman. *An RDF Query and Transformation Language*, pages 23–39. Semantic Web and Peer-to-Peer. Springer Berlin Heidelberg, 2006.
4. John Davies, Richard Weeks, and Uwe Krohn. QuizRDF: Search technology for the Semantic Web. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, pages 112–119. BTexact Technologies, IEEE Computer Society, 2004.
5. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 2000.
6. Tim Finin, James Mayfield, Clay Fink, Anupam Joshi, and Scott R. Cost. Information Retrieval and the Semantic Web. In *Proceedings of the 38th International Conference on System Sciences*, 2005. Received Best mini-track paper award.
7. Christiaan Fluit, Marta Sabou, and Frank Harmelen van. Ontology-based information visualisation: Towards semantic web applications. Springer Verlag, 2005.
8. Thomas M.J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.
9. E Miller. Digital libraries and the semantic web. http://www.w3.org/2001/09/06-ecdl/slide1-0.html, 2001. Accessed 11 December 2006.
10. Paul Mutton and Jennifer Golbeck. Visualization of semantic metadata and ontologies. In *Seventh International Conference on Information Visualization (IV03)*, pages 300–305. IEEE, 2003.
11. Bijan Parsia, Taowei Wang, and Jennifer Golbeck. Visualizing web ontologies with cropcircles. In *End User Semantic Web Interaction WS*. ISWC 2005, 2005.
12. D.A. Quan and David R. Karger. How to make a semantic web browser. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 255–265, New York, NY, USA, 2004. ACM Press.
13. Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Visual Languages*, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.