15

The Role of Explicit Semantics in Search and Browsing

Michiel Hildebrand, Jacco van Ossenbruggen and Lynda Hardman
CWI, Amsterdam, The Netherlands

In recent years several Semantic Web applications have been developed that support some form of search. These applications provide different types of search functionality and make use of the explicit semantics present in the data in various ways. The aim of this chapter is to give an overview of the semantic search functionalities provided by the current state of the art in Semantic Web applications. Excluding search based on structured query languages such as SPARQL (Prud'homme and Seaborne 2007), we focus on queries based on simple textual entry forms and queries constructed by navigation (e.g. faceted browsing). We provide a systematic understanding of the different design dimensions that play a role in supporting search on Semantic Web data. Using the required insights, we describe the design decisions made in two of our own tools with practical use cases.

Section 15.1 establishes the basic search terminology used throughout the chapter. Section 15.2 provides an in-depth analysis of the different types of search functionality based on a survey of 35 Semantic Web applications. We argue that search is far from trivial, and list the different roles semantics currently plays throughout the various phases of the search process. Then we show with two use cases how 'semantic' functionality can support the user while searching for museum objects in Section 15.3 and use faceted navigation to explore a collection of news images in Section 15.4. We summarize our main conclusions in Section 15.5.

15.1 Basic Search Terminology

We introduce the basic definitions used throughout the paper.

Multimedia Semantics: Metadata, Analysis and Interaction, First Edition. Edited by Raphaël Troncy, Benoit Huet and Simon Schenk.

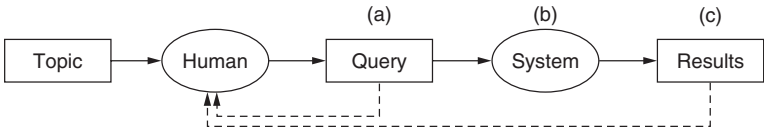

Figure 15.1 High level overview of text-based query search: (a) query construction; (b) search algorithm of the system; (c) presentation of the results. Dashed lines represent user feedback. Figure adapted from TRECVID (2008)

Search process. We follow the TRECVID 2007 Evaluation Guidelines (TRECVID 2008) and divide the search process into three different phases: query construction, execution of the core search algorithm and presentation and organization of the results. We also take into account user feedback on the query and on the results (see also Figure 15.1).

Semantic search. We use the term 'semantic search' when semantics is used during any of the phases in the search process.

Type of results. Traditional search engines on the web typically assume the result of a search to be a set of web resources (e.g. text documents, images or video). This also holds for some Semantic Web applications. Others, however, return sets of matching URIs, sets of matching triples (an RDF sub-graph) or a combination of these. Often the behavior of a system depends on the type of result it assumes, so we make this explicit wherever this is relevant.

Overview pages and **surrogates**. We refer to presentation of the (k first) results as the *overview page*, which typically represent results using *surrogates*. For example, resulting HTML pages can be represented by their title and a text snippet, while image or video results are often represented by thumbnails.

Local view page. Surrogates typically provides links to the full presentation of the result they represent. The latter is either the full presentation of an information resource (e.g. the full HTML page that is linked from the snippet presented on the overview page), some human-readable representation of metadata associated with the result, or a human-readable representation of a resulting subgraph. Following Rutledge et al. (2005), we refer to the latter presentation as a *local view page*.

Syntactic matching. Syntactic search matches the query against the textual content of the resources (if applicable), the literals in the RDF metadata, the URIs in the system, or a combination of these.

Semantic matching. We use the term 'semantic matching' for those algorithms that, in addition to syntactic matching, use the graph structure of the RDF metadata and/or its semantics to find results.

15.2 Analysis of Semantic Search

We systematically scanned all proceedings of the International and European Semantic Web Conference series as well as the *Web Semantics* journal, to compile a list of end-user applications described or referred to. For each system we collected basic characteristics such as the intended purpose, intended users, the scope, the triple store and the technique or software used for literal indexing. The resulting document was made available online[1]

[1] http://www.webscience.org/swuiwiki?title = Semantic_Search_Survey

Table 15.1 Functionality and interface support in the three phases of semantic search

Query construction	
Feature: **Functionality**	**Interface Components**
Free text input: Keywords, natural language	Single text entry, property-specific fields
Operators: Boolean constructs, syntactic disambiguation, semantic constraints on input/output	Application-specific syntax
Controlled terms: Disambiguate input, restrict output, select predefined queries	Value lists, faceted browser, graph
User feedback: Pre-query disambiguation	Autocompletion
Search algorithm	
Syntactic matching: Exact, prefix or substring match, minimal edit distance, stemming	Not applicable
Semantic matching: graph traversal, query expansion, spread activation, RDFS/OWL reasoning	Not applicable
Result presentation	
Data selection: Selected property values, class-specific template, display vocabularies	Visualized by text, graph, tagcloud, map, timeline, calendar
Ordering: Content and link structure based ranking	Ordered list
Organization: Clustering by property, by result path or dynamic	Tree, nested box structure, clustermap
User feedback: Post-query disambiguation, recommendation of related resources	Facets, tagcloud, value list

and announced on three public mailing-lists.[2] Additionally, we sent personal emails to the authors of papers and developers of all systems included. This resulted in an updated version of the online document. This update was based on 15 email threads, in which additional information was provided for 11 systems and 6 additional systems within the scope of the survey were recommended, giving a total of 35 systems.

Based on the data resulting from the survey, we perform a more thorough analysis of the three individual phases in the search process of Figure 15.1. For each of these we consider the underlying functionality and the features of the corresponding user interface. The results of this analysis are summarized in Table 15.1, and the table also provides the structure of the remainder of this section. We discuss query construction in Section 15.2.1, the search algorithms in Section 15.2.2 and the presentation of the results in Section 15.2.3. Note that the examples and references merely serve as illustrations; the full analysis with references is available in the online survey.

15.2.1 Query Construction

The search process starts with the user constructing a query that reflects his or her information needs. We describe the functionality for this process as provided by the systems in the survey and how this functionality is supported at the interface.

[2] `public-xg-mmsem@w3.org`, `semantic-web@w3.org`, `public-semweb-ui@w3.org`

Functionality

Constructing a query in free text requires little knowledge of the system and data structure. The price users have to pay is ambiguity: words can have multiple meanings (lexical ambiguity) and a complex expression can have multiple underlying structures (structural ambiguity). Ambiguous input often leads to irrelevant search results. To reduce ambiguity several systems allow additional query constructs beyond free text input: structural and semantic operators and controlled terms. We describe free text input and the additional query constructs and the role of user feedback in the process of matching free text with controlled terms.

Free text input is supported in existing systems in three ways. First, full text search allows the user to find all resources with matching textual content or metadata. In many semantic search engines full text search is the main entry point into the system (Celino et al. 2006; DERI 2005; Ding et al. 2005; ;Guha et al. 2003; Schreiber et al. 2006). Second, free text input can be restricted to match a value of a specific property. In faceted browsers this is the case when searching for a value within a particular facet (Hildebrand et al. 2006; Schraefel et al. 2005; SIMILE 2003). Finally, systems such as AquaLog (Lopez et al. 2005) and Ginseng (Bernstein et al. 2006) support free text input in the form of natural language expressions.

Syntactic operators explicitly define the interpretation of complex search terms. Well-known examples are the boolean operators AND and OR. Several applications employ third-party search libraries such as Apache Lucene,[3] which typically provide an extensive collection of syntactic control structures.

Semantic operators add explicit meaning to a query. In SemSearch, for example, the user specifies to which RDFS or OWL class a result should belong. The authors illustrate this with the example `article:motta` for which the system retrieves all resources that are of type `article` and match the search term `motta` (Lei et al. 2006).

Controlled terms provide the use of predefined concepts. In QuizRDF (Davies and Weeks 2004) the user selects an RDF class to determine the type of the search term. Other systems provide autocompletion to support users with keyboard-based input of controlled terms (Hildebrand et al. 2006; Hyvönen and Mäkelä 2006; Kiryakov et al. 2004; Schraefel et al. 2005; SIMILE 2003). A different approach is seen in DBin (Tummarello et al. 2006) and Haystack (Quan and Karger 2004), which allow the user to select predefined queries.

User feedback on the input is useful when there are multiple controlled terms that match with the free text input. Several systems allow the user to select the intended term before it is processed by the search algorithm (Celino et al. 2006; Hildebrand et al. 2006; Hyvönen and Mäkelä 2006). This form of user feedback allows pre-query disambiguation. In contrast, post-query disambiguation is performed on the results of the search algorithm.

Interface

The basic interface components to enter or construct a query are text entry boxes and value selection lists. These components are used in various designs, of which we mention three. If applicable, we describe the link from the interface components to the underlying

[3] `http://lucene.apache.org/java/docs/queryparsersyntax.html`

data structure. Furthermore, we describe the interface aspects of user feedback on the input and several proposals for more advanced query construction.

A single text entry field is sufficient for free text input, e.g. Google and several systems that we analyzed (Celino et al. 2006; Davies and Weeks 2004; Ding et al. 2004; Guha et al. 2003; Schreiber et al. 2006). Additional features included by some systems are selectable result types (Ding et al. 2004) and options for the search algorithm or presentation of the results (Davies and Weeks 2004).

Property-specific search fields support query construction guided by a specific set of possible search values (Heflin and Hendler 2000; Kiryakov et al. 2004; Metaweb 2007; Mika 2006). The value sets are typically defined by the range of the corresponding RDF property.

Faceted browsing allows the user to constrain the set of results within a particular facet. Typically, facets are directly mapped to properties in RDF. Alternatively, the mapping is made by projection rules. The advantage of an indirect mapping is that this allows the developer to define facets that match the user's needs while keeping the data structure unchanged (Suominen et al. 2007). Faceted browsing is applied to Semantic Web data by various authors (Fluit et al. 2003; Hildebrand et al. 2006; Hyvönen et al. 2005; Oren et al. 2006; Schraefel et al. 2005; SIMILE 2003; Suominen et al. 2007) as well as by the company Siderean[4] in the Seamark Navigator.

User feedback is typically provided after the query has been entered, or dynamically during the construction of the query as a form of *semantic autocompletion*. The former method is used in Squiggle (Celino et al. 2006) and MuseumFinland (Hyvönen et al. 2005) where the disambiguation of the matching query terms is presented after submitting the query. In semantic autocompletion the system suggests controlled terms with a label prefix that matches the text already typed in. Hyvönen and Mäkelä (2006) describe the idea of semantic autocompletion and several implementations. In faceted interfaces autocompletion is often used within a single facet (Hildebrand et al. 2006; Kiryakov et al. 2004; Schraefel et al. 2005; SIMILE 2003).

In general there is a clear need for simple interfaces, such as the single text entry field. On the other hand, interface designs that support more complex interaction styles potentially give the user more control, which is useful for the formulation of more precise information needs.

15.2.2 Search Algorithm

All text-based search involves some form of syntactic matching of the query against textual content and/or metadata, an aspect well covered in information retrieval (Baeza-Yates and Ribeiro-Neto 1999). Semantic matching can extend syntactic matching by exploiting the typed links in the semantic graph. Before we describe semantic matching we briefly describe syntactic matching, focusing on the indexing functionality and the support that is already provided by the low-level software on which various systems are built.

[4] http://www.siderean.com/

Syntactic Matching

All systems in our study index the textual data in their collection for performance reasons. Which textual data is indexed (e.g. the content, the metadata or the URIs) is important for the search functionality of the system. In an ontology search engine such as Swoogle, users might want to search on URIs (Ding et al. 2005). In annotated image collections the metadata forms the primary source for indexing (Celino et al. 2006; Hyvönen et al. 2005; Schreiber et al. 2006). For images that occur in web pages the contextual text provides an alternative source (Celino et al. 2006; Falcon-S 2005). Indices can be based on the complete word or on a stemmed version. Some interface functionalities require additional features. Autocompletion interfaces, for example, require efficient support for prefix matching. Some triple stores provide built-in support for literal indexing, for example, OpenLink Virtuoso[5] and SWI Prolog's Semantic Web library.[6] Alternatively, a search engine, such as Lucene, can be used together with a triple store.

Semantic Matching

After syntactic matching, the structure and formal semantics of the metadata can be used to extend, constrain or modify the result set. Note that in a connected RDF graph, any two nodes are connected by a path. Naive approaches to semantic search are computationally too expensive and dramatically increase the number of results. Systems thus need to find a way to reduce the search space and to determine which semantically related resources are really relevant.

Inspired by the semantic continuum described by (Uschold 2003), we distinguish three levels of semantic matching: graph traversal, explicit use of thesaurus relations and inferencing based on the formal semantics of RDF, RDFS and OWL.

Graph traversal takes only the structure of the graph into account. Several techniques are in use to constrain graph search algorithms. In Tap, constraints define which relations to traverse for the instances of a particular class (Guha et al. 2003). Alternatively, a weighted graph search algorithm may constrain the possible path structures and path length. Such an algorithm requires the assignment of weights to the edges in the graph, where the weights reflect the importance of the corresponding RDF relations. In e-Culture, weights are manually assigned to RDF relations (Schreiber et al. 2006). SemRank automatically computes weights based on statistics derived from the graph structure (Anyanwu et al. n.d.). Spread activation (Rocha et al. 2004) is another computationally attractive technique for graph traversal, which can incorporate weights as well as the number of incoming links.

Thesaurus relations are sometimes used for query expansion. With the acceptance of SKOS (Miles et al. 2005) as a standard representation for thesauri, semantic matching with hierarchical broader term, narrower term, and the associative related term can be implemented in a generic way. The Squiggle framework is an example in which this is done (Celino et al. 2006). Facet browsers typically rely on hierarchical thesaurus relations to restrict their result sets (Hildebrand et al. 2006; Hyvönen et al. 2005). Within the FACET project the integration of thesauri in the search process is studied extensively. This resulted in a demonstrator as well as a proposal for a semantic expansion

[5] http://www.openlinksw.com/

[6] http://www.swi-prolog.org/packages/semweb.html

service (Binding and Tudhope 2004), which in turn formed the basis for the experimental SKOS API.[7]

RDFS/OWL reasoning can also influence the search results. Several systems support RDFS subsumption once an RDF class is selected in the interface (Auer et al. 2006; Duke et al. 2007; Lei et al. 2006; Tummarello et al. 2006). In Dose, specialization and generalization over the subclass hierarchy are used dynamically according to the number of search results (Bonino et al. 2004). Some systems support partial OWL reasoning, and process, for example, only the OWL identity relations. In Flink (Mika 2005) and SWSE (DERI 2005) these are extensively used to model the identity between extracted entities.

15.2.3 Presentation of Results

We describe how explicit semantics is used to extend the baseline functionality in the presentation of search results, and the techniques that are used to visualize the results in the interface. As a baseline we consider the presentation of search results by popular search engines such as Google: the information selected for presentation is the URI or label of the result, surrogates of the content (e.g. text snippets or image thumbnails) and, optionally, additional information such as the file size. The results are typically organized in a plain list and ordered by relevance. We describe, for each aspect, how additional semantics are used to extend this baseline.

Functionality

We consider three aspects of the presentation: selecting what data to present, organizing the results and ordering the results. In addition, we discuss the function of user feedback on the results.

Selecting what to present is tightly bound with what the search engine considers to be a 'result'. If the result is a Web page or other information resource, traditional surrogates are typically used. When the search result is a set of URIs referring to nodes in an RDF graph, or a set of RDF triples, systems need to invent new ways to represent the results in their overview page. In most systems we studied, the decision on what (meta)data is used for the surrogates is hardwired into the system. QuizRDF supports template definitions for each RDF class (Davies and Weeks 2004). Dbin (Tummarello et al. 2006) creates templates for specific user tasks and domains. Display vocabularies such as Fresnel (Bizer et al. 2005), as used by Longwell (SIMILE 2003), provide full control over what data to select for presentation and how to present it.

Turning to *organizing the results*, semantics can also play a role in grouping semantically similar results together in the presentation, a feature commonly referred to as *clustering*. Assuming that users are interested in the results of only one cluster, clustering can also be considered as a form of post-query disambiguation. In our study we found several forms of clustering. In many systems, the values of a particular property are used to group the result set on common characteristics within a particular dimension. In Guha et al. (2003) results with similar types are clustered together. In faceted browsers similar behavior is found: systems described in SIMILE (2003), Hildebrand et al. (2006), and

[7] http://www.w3.org/2001/sw/Europe/reports/thes/skosapi.html

Hyvönen et al. (2005) all support clustering on the values of a particular facet. Noadster uses concept lattices to determine dynamically which properties to use for a given result set (Rutledge et al. 2005). In e-Culture (Schreiber et al. 2006) the RDF path between the literal that is syntactically matching the query and the result may span more than one property. Clustering the results on these paths illustrate the interpretations of the query.

The *ordering of results* can be done using different techniques. Ranking of results based on relevance is covered well in information retrieval (Baeza-Yates and Ribeiro-Neto 1999). Numerous algorithms have been developed, evaluated and applied in successful applications. Term frequency-inverse document frequency (tf-idf) is a syntactic measure often used to determine the importance of a word based on the number of occurrences in a document relative to the number of occurrences in the entire collection. Many systems in our study use Lucene, which provides ranking based on tf-idf. In addition to textual content, the link structure is another source for ranking. Swoogle uses a variant of PageRank (Page et al. 1998) to measure the relevance of RDF documents. PageRank was adapted to compensate for different types of relations that link RDF documents and terms (Ding et al. 2005). In SWSE (Hogan et al. 2006) a variant of PageRank based on the principle of focused subgraphs (Kleinberg 1999) is used.

In our study, we did not find instances where matching and ranking algorithms are influenced by *user feedback*. We mainly encountered user feedback in disambiguating, specializing, generalizing or expanding the result set. Most systems support expansion of a query by adding a keyword or by selecting a value from a property field or facet. In several systems, post-query disambiguation of free text input is supported through the selection of an RDF type (Berlin 2007; Davies and Weeks 2004; DERI 2005; Duke et al. 2007). Alternatively, queries can be specialized or generalized with concepts from narrower or broader thesaurus relations (Celino et al. 2006; Hildebrand et al. 2006; Hyvönen et al. 2005). An unwanted side effect of query refinement is the risk of ending up with no results. This can be avoided by restricting the user beforehand to the use of only those terms that lead to results. This is one of the principles behind faceted browsing interfaces (Yee et al. 2003).

We observed that domain-specific applications use the semantics to organize the search results into clusters. Domain-independent search engines typically rely on ranking techniques for effective presentation of the search results.

Interface

Most systems provide a straightforward interface that directly reflects the structure of the data selected and how it is organized. Typical examples include numbered lists for a linearly ranked set of results or visual grouping of clustered results in nested box layout structures. Since RDF is represented as a graph, visualizing the data as a graph may seem a straightforward choice. However, from a user interface perspective, 'big fat graphs' quickly become unmanageable (Schraefel and Karger 2006), with only a few exceptions, including the visualization of social networks between small groups of people (Mika 2005). Other visualization techniques (Geroimenko and Chen 2003b) we encountered include the following:

- *Tagclouds* that indicate the importance of textual metadata with variations in the font size. OpenAcademia (Mika 2006) visualizes the concepts related to research publications and search. DBpedia.org (Berlin 2007) presents the available RDF types of the search results.
- *Clustermaps* that visualize the overlap between classes of instances, without needing an explicit concept representing this overlap (Geroimenko and Chen 2003a). For example, in AutoFocus a clustermap visualizes the results of individual constraints as well as result sets that satisfy multiple constraints (Fluit et al. 2003).
- *Data type-specific* visualizations are used in several systems to present space and time on a geographical map, timeline or calendar. The Simile timeline,[8] several map visualization tools and Google Calendar can be used through publicly available APIs. Hence, we do not list the individual systems that make use of these.
- *Local view pages* provide a detailed presentation of the metadata associated with single URI. Systems such as Tabulator (Berners-Lee et al. n.d.) and Disco (Bizer and Gauss 2007) are based on the notion of a *concise bounded description* (CBD)[9] and present the statements where the current focus URI is a subject. Others systems' local view pages may contain all statements in which the URI occurs either as a subject or object. Sesame's URI explorer pages,[10] Noadster (Rutledge et al. 2005) and E-culture (Schreiber et al. 2006) also include statements where the URI plays the role of the property.

15.2.4 Survey Summary

The survey shows that explicit semantics can play a role in all three parts of the search process. In the query construction controlled terms are used to create unambiguous queries. Terms are either presented in a selection list or made available on demand by autocompletion. Faceted interfaces give the user even more control, allowing the user to define the particular role of a query term. Natural language input also allows the user to construct more precise queries. In this case the system has to do the interpretation of the query.

Current search applications make only limited use of OWL reasoning. Only equivalence reasoning is used in several applications. Lightweight reasoning, based on thesaurus relations, is a more common feature in search algorithms. The SKOS relations for broader term and related term are used both for query expansion as well as to recommend related results. More generic graph search algorithms have the advantage that all relations in the data can be considered at the cost of returning less relevant results.

In the result presentation standard representations of space and time properties make visualizations on a map, timeline or calendar a common feature. The link structure of the graph provides valuable information for ranking algorithms as well for clustering algorithms. Determining the effectiveness of these techniques remains, however, unclear. Note that, for example in information retrieval, there is a long tradition of evaluating

[8] http://simile.mit.edu/timeline/
[9] http://www.w3.org/Submission/CBD/
[10] http://www.openrdf.org/

the quality of retrieval systems. Conference series such as TREC and INEX contribute to an (evolving) community consensus about which dimensions to evaluate, and how to measure a system's performance on that dimension. It is safe to say that within the Semantic Web community, we have not yet developed a similar consensus about the use of explicit semantics to improve search, and how to evaluate and to compare semantic search systems.

We now turn to a detailed description of two types of semantic search supported by our own tool, ClioPatria (Wielemaker et al. 2008). We first describe semantic keyword search with a cultural heritage use case. Then we describe how faceted navigation can be used for the exploration of a collection of news images.

15.3 Use Case A: Keyword Search in ClioPatria

The search facility of the MultimediaN E-Culture demonstrator[11] allows the user to search for museum objects that are semantically related to one or more keywords. A typical scenario for this type of semantic search is the exploitation of vocabulary alignments. Consider a use case in which the user is interested in finding works from the Japanese Tokugawa style period. We describe the search process when using the E-Culture demonstrator to find these artworks.

15.3.1 Query Construction

To support keyword search a simple Google-like interface is sufficient. The user types the keyword 'Tokugawa' into the text entry field and submits this query. Instead of submitting a keyword query, the user can choose to first disambiguate the query. The system uses autocompletion to suggest artworks or thesaurus terms while the user is typing. Figure 15.2 shows the suggestions given when the user typed 'toku'. The suggestions are organized into different clusters, grouping similar suggestions together.

15.3.2 Search Algorithm

Consider the situation in Figure 15.3, which is based on real-life data. The user has submitted the query 'tokugawa'. The Dutch ethnographic museum in Leiden actually has works in this style in its digital collection, such as the work shown on the right of the figure. However, the Dutch ethnographic thesaurus SVCN, which is being used by the museum for indexing purposes, only contains the label 'Edo' for this style. Fortunately, another thesaurus in our collection, Getty's Art and Architecture Thesaurus[12] (AAT), does contain the same concept with the alternative label 'Tokugawa'.

To exploit all possible relations available in the heterogeneous data a generic graph traversal algorithm is used. Either the keyword query or the disambiguated thesaurus term is the starting point of the graph traversal.

Figure 15.4 shows the result of a graph search for the keyword query 'tokugawa'. The three rectangular boxes on the left show the literals that matched the keyword. The

[11] http://e-culture.multimedian.nl/demo/search

[12] http://www.getty.edu/research/conducting_research/vocabularies/aat/

Figure 15.2 Autocompletion suggestions are given while the user is typing. The partial query 'toku' is contained in the title of three artworks, there is one matching term from the AAT thesaurus and the artist Ando Hiroshige is found as he is also known as Tokubel

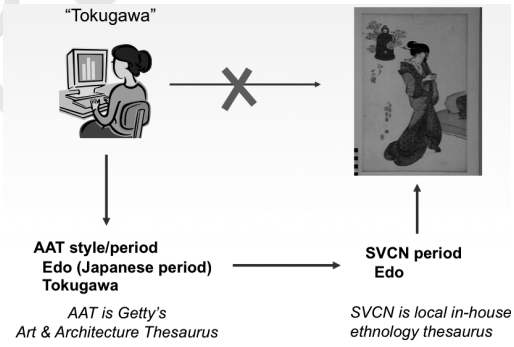

Figure 15.3 A user searches for 'tokugawa'. The Japanese painting on the right matches this query, but is indexed with a thesaurus that does not contain the synonym 'Tokugawa' for this Japanese style. Through a 'same-as' link with another thesaurus that does contain this label, the semantic match can be made

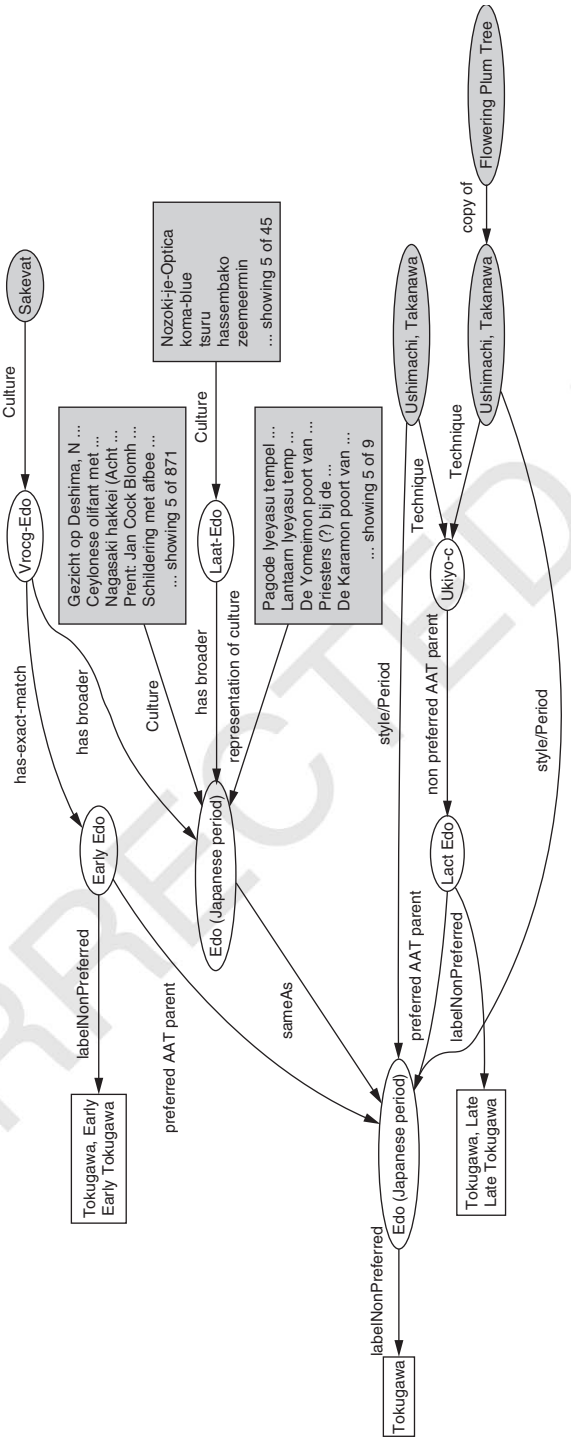

Figure 15.4 Result graph of the E-Culture search algorithm for the query 'Tokugawa'. The rectangular boxes on the left contain the literal matches, the colored boxes on the left contain a set of results, and the ellipses a single result. The ellipses in between are the resources traversed in the graph search

algorithm found that these literals are labels of three thesaurus terms. All three are from the AAT thesaurus. Note that the three terms are related. The terms 'Early Edo' and 'Late Edo' are specializations of the Japanese period 'Edo'. Let us follow the generic term 'Edo' two steps further. Through an `owl:sameAs` relation the algorithm has found the equivalent term from the Dutch ethnographic thesaurus. This term leads us to many museum objects, the first five of which are shown in each of the rectangular boxes on the right of the figure. The AAT term 'Edo' leads to other interesting results. For two paintings the term was used for the 'Style/Period' property. Both are painted by the famous Japanese artist Ando Hiroshige, titled *Plum Estate* and *Ushimachi*. Note that one of these paintings is also found because it is a woodblock print (*ukiyo-e*), which is as specialization of the 'Late Edo' style. This print also leads to another object, namely *Flowering Plum Tree* by Vincent van Gogh. This painting is found because van Gogh made a copy of the Japanese original.

The entire graph contains some RDFs and OWL properties (or subproperties thereof). Most properties are, however, domain-specific properties used within the thesauri or to index the museum objects. The generic traversal algorithm allows the system to exploit all these properties.

15.3.3 Result Visualization and Organization

Different paths in the search graph indicate different semantic relations. In other words, a path reflects an interpretation of the query. In the E-Culture demonstrator the paths are, therefore, used to disambiguate the search results. Figure 15.5 shows the presentation of

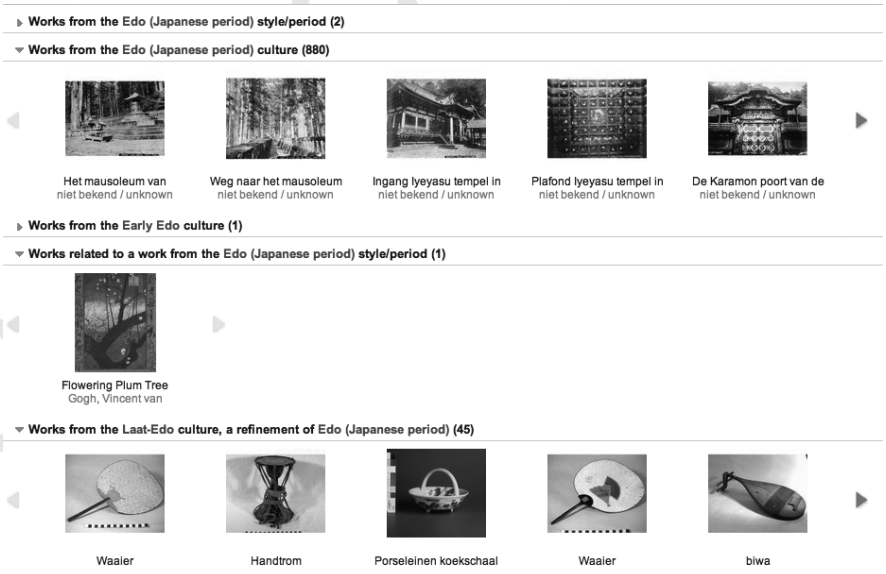

Figure 15.5 Presentation of the search results for the query 'Togukawa' in the E-Culture demonstrator. The results are presented in five groups (the first and third groups have been collapsed). Museum objects that are found through a similar path in the graph are grouped together

the search results for the query 'Tokugawa'. The museum objects with a similar search path are grouped together. Five clusters are shown. For readability the first and third clusters have been collapsed. An additional abstraction step is performed to merge similar clusters together. In this step the resources in the graph path are abstracted over the subproperty and subclass relations. The seven result groups (colored boxes on the right) in the graph of Figure 15.4 are merged into five, because the property 'representation of culture' is a subproperty of 'culture'. The results 'Ushimachi, Takanawa' and 'Plum Estate, Kameido' share the same shortest path.

15.4 Use Case B: Faceted Browsing in ClioPatria

Within K-Space, ClioPatria is used to support search and browsing of news items. These news items are described with multimedia standards, news codes from the IPTC standard and additional metadata from various thesauri. The additional metadata is acquired through extraction of named entities such as persons, organizations and locations, from the textual stories. The extracted named entities are mapped to existing resources available on the Web, such as locations from Geonames, and individuals from DBpedia. The dataset uses in this demonstration consists of news items from 2006, including the soccer World Cup.

Consider a use case in which the user wants to selects a number of images that reflect different aspects of the French soccer player Zinedine Zidane. Keyword search allows us to find a set of news items related to Zidane, but for a more thorough exploration of these results additional control is required. Faceted presentation of the metadata gives the user additional control in the formulation of her request. The facets allows the user to specify relatively complex queries by simply following navigation links.

15.4.1 Query Construction

As in the previous use case, we start with a simple keyword search to find news items that are related to 'zidane'. The results contain both textual news messages as well as photos. Among the results there are news items with the keyword 'zidane' in the title/headline, photos showing Zidane, photos with his team mates and many photos with happy as well as sad fans. On the result page we can now activate the faceted navigation component and use it to further explore the result set.

A screenshot of the faceted interface from ClioPatria is shown in Figure 15.6. The top part contains four facets: `document type`, `creation site`, `event` and `person`. The result viewer, visible below the facets, contains news items related to the keyword 'zidane'. The current query is shown in the header of the result viewer. The user can extend the query by selecting values from the facets. In this case the value 'photo' is selected from the `document type` facet. The other facets only contain values that correspond to the current result set. Note that this prevents the user from constructing queries that lead to an empty result set.

The person facet contains, besides Zidane himself, other persons who are related to the photos. Most of the persons are soccer players, but the list also contains the former French president Jacques Chirac. Selecting the value corresponding to this person constrains the result set to only those images related to both Zidane and Chirac. In this case the photos

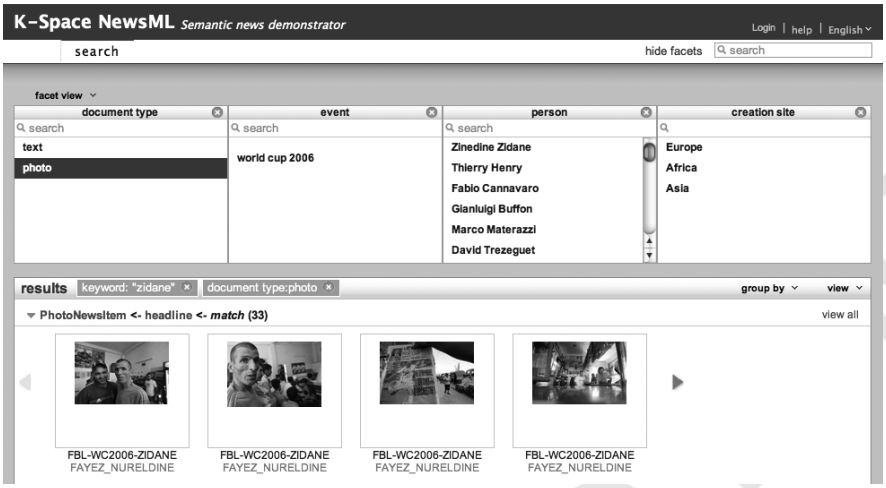

Figure 15.6 Faceted interface of the NewsML demonstrator. Four facets are active: `document type`, `creation site`, `event` and `person`. The value 'photo' is selected from the `document type` facet. The full query also contains the keyword 'Zidane', as is visible in the header above the results

Figure 15.7 Hierarchical organization of the values in the `creation site` facet. The value 'Europe' is selected and below it the four countries in which photos related to Zidane are created

show Zidane visiting the French president after the lost final of the 2006 World Cup. Clicking on a facet value that is selected will deselect this value and remove it from the query.

The screenshot in Figure 15.7 shows the `creation site` facet in which the value 'Europe' is selected and below it the four European countries in which the photos are created. We can navigate further into this hierarchy by selecting one of the countries. Selecting the value 'France' constrains the result set to photos of soccer fans in different regions of France. Selecting the value 'Germany' constrains the result set to all cities where Zidane played during the World Cup. By selecting the city Berlin, the result set now only contains photos of the World Cup final. Finally, we constrain the result set to photos of the infamous head-butt incident by selecting the Italian player 'Marco Materazzi' from the `person` facet.

15.4.2 Search Algorithm

The values selected from a facet constrain the results to those news items that are indexed with that particular value. Each additional constraint adds a new conjunct to this this query. Values selected from a hierarchical facet require an additional step, as the news items may not be directly indexed with a value selected from the hierarchy. A news item is considered a result when the value it is indexed with is reachable from the selected value through some relation. In a SKOS thesaurus this relation corresponds to `skos:broader` or `skos:narrower`.

To support the combination of semantic keyword search and faceted navigation, the graph search algorithm has to be combined with facet constraints. In ClioPatria this is solved by executing the graph search and using the facet constraints to determine if a resource encountered during the search is a result.

15.4.3 Result Visualization and Organization

After the initial query of photos related to Zidane the `creation site` facet contained three continents: Europe, Africa and Asia. To get a view on the photos for each continent

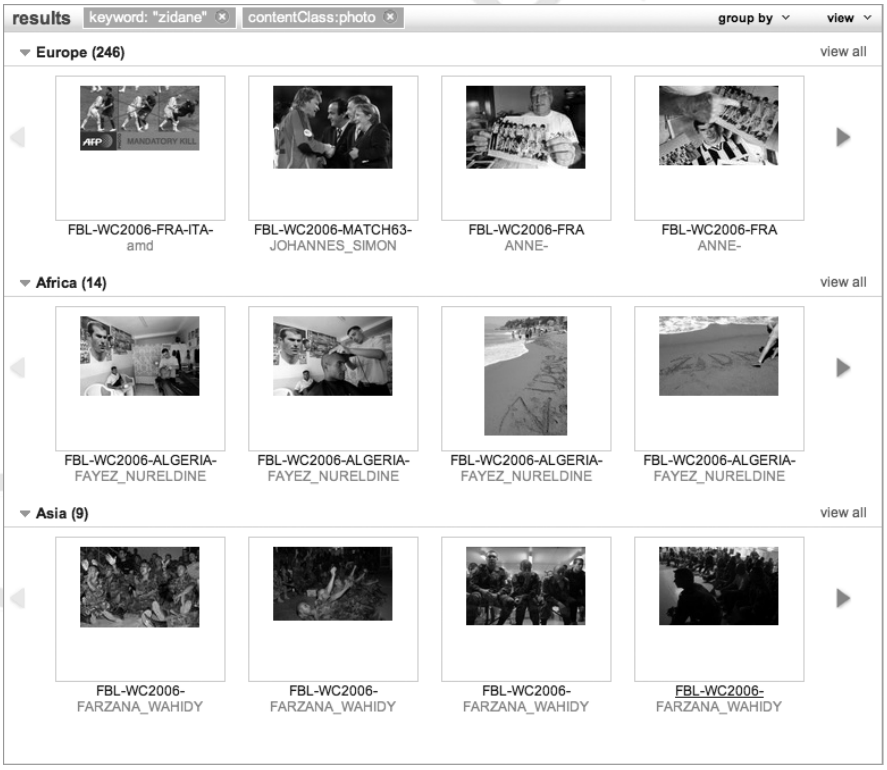

Figure 15.8 Grouped presentation of search results. The photos related to Zidane are presented in groups with the same creation site

each facet value can be selected in turn, but to compare different groups of photos it is much more convenient to group the results. The grouping can be selected from the drop-down menu in the result viewer. The screenshot in Figure 15.8 illustrates that the photos made in Europe primarily show soccer matches and French supporters, the photos in Africa show Zidane's home country Algeria, and the photos in Asia depict French soldiers stationed in Afghanistan while watching the World Cup on television.

The organization and visualization of the facet values is important as well. The hierarchical organization of the values in the `creation site` facet provides different geographical perspectives on the data, by continent, country, region or city. An alternative organization is to group similar facet values together. For example, grouping persons by their nationality would allow the user to select all news items related to German persons. The visualization of facet values is important when there are ambiguous terms. Instead of presenting just the label, further information can be added. For example, for persons we can present the nationality and birth date alongside the person's name.

15.5 Conclusions

We have analyzed the state of the art in search as currently implemented in Semantic Web applications. We have identified the various roles played by semantics in query construction, the core search algorithm and presentation of search results. Our study shows that many systems already support aspects of semantic search. For some, it is even the main entry point into the underlying data. The quality of the search functionality and its user interface thus has a significant impact on the overall usability of these applications. Working toward generic libraries and publicly available services that support the individual processes will help developers to incorporate more semantic search functionality into applications. To improve the uptake of Semantic Web applications by end users, we feel our community should strive to make this research area more mature and start to develop methods for objective and systematic evaluation of the functionality and interface aspects of end-user applications.