End-user support for access to heterogeneous linked data

Hildebrand, M.

Chapter 2

Related work

In this chapter we identify different types of user-oriented search functionality and presentation methods to access Semantic Web data. We summarise our literature study of Semantic Web applications for search and browsing. The basic terminology is introduced and the different features found in the literature are analysed in the form of a survey. From this analysis, we identify a number of problems with user-oriented support for accessing Semantic Web data. Some of these are explored in the case studies in Chapters 3, 4 and 5, where we design and evaluate support for a number of the specific features discussed.

This chapter is based on a survey conducted in May 2007 and will appear as the chapter "The role of explicit semantics in search and browsing" in the book Multimedia Semantics: Metadata, Analysis and Interaction (Hildebrand et al. 2010). It is co-authored by Jacco van Ossenbruggen and Lynda Hardman. Since the original survey, the area of semantic search has grown and Web companies are starting to integrate semantic technologies into their search engines (Hendler 2010). In particular, advances have been made in the extraction of semantic relations using natural-language processing. This is, however, outside the scope of this thesis. In addition, we observe the uptake of semantic relations to improve the presentation of the search results, e.g. to provide more informative results and/or organise the results. Several of these strategies were already investigated in the prototypes discussed in this chapter.

2.1 Introduction

In recent years several Semantic Web applications have been developed that support some form of search. These applications provide different types of search

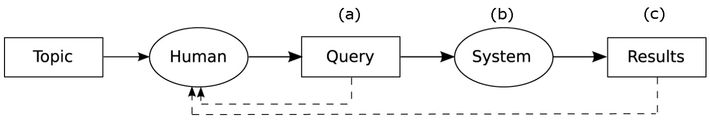

Figure 2.1: High level overview of text-based query search: (a) query construction, (b) search algorithm of the system, (c) presentation of the results. Dashed lines represent user feedback. Image based on http://www-nlpir.nist.gov/projects/tv2007/.

functionality and make use of the explicit semantics present in the data in various ways. The goal 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 (W3C 2006), 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.

The next section establishes the basic search terminology used throughout the chapter. Section 2.3 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 play throughout the various phases of the search process. We discuss the main conclusions in section 2.4.

2.2 Basic Search Terminology

We introduce the basic definitions used throughout the chapter.

Search process We follow the TRECVID 2007 Evaluation Guidelines (TRECVID organizers 2007) and divide the search process into three different phases: query construction, execution of the core search algorithm and presentation and organisation of the results. We also take into account user feedback on the query and on the results (see also Figure 2.1).

Semantic search We use the term semantic search when semantics are 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 objects, 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 behaviour 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 provide links to the full presentation of the result they represent. The latter is either the full presentation of an information object (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 sub-graph. 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 objects (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 also use the graph structure of the RDF metadata and/or its semantics to find results.

2.3 Analysis of semantic search

We systematically scanned all proceedings of the International and European Semantic Web Conference series as well as the Journal of Web Semantics, 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 that is used for literal indexing. The resulting document was made available online[1] and announced on three public mailing-lists[2]. Additionally, we sent personal emails to the authors of papers and developers of all included systems. 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 performed a more thorough analysis of the three individual phases in the search process of Figure 2.1. For each of these we consider the underlying functionality and the features of the corresponding user interface. The results of this analysis are summarised in Table 2.3,

[1] http://swuiwiki.webscience.org/index.php/Semantic_Search_Survey
[2] public-xg-mmsem@w3.org, semantic-web@w3.org, public-semweb-ui@w3.org

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, RDF-S/OWL reasoning	Not applicable

Result presentation

Data selection	Selected property values, class-specific template, display vocabularies	Visualised by text, graph, tag-cloud, map, timeline, calendar
Ordering	Content and link structure based ranking	Ordered list
Organisation	Clustering by property, by result path or dynamic	Tree, nested box structure, clustermap
User feedback	Post-query disambiguation, recommendation of related objects	Facets, tagcloud, value list

Table 2.1: Functionality and interface support in the three phases of semantic search.

and the table also provides the structure of the remainder of this section. We discuss query construction in section 2.3.1, the search algorithms in 2.3.2 and the presentation of the results in 2.3.3. Note that the examples and references merely serve as illustrations, the full analysis with references is available in the online survey.

2.3.1 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.

Functionality Constructing a query in free text requires little knowledge of the system and the 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 objects with matching textual content or metadata. In many semantic search engines full text search is the main entry point into the system (Ding et al. 2005; Schreiber et al. 2006; Celino et al. 2006; Guha et al. 2003; DERI 2007). 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 (SIMILE 2005; m.c. schraefel et al. 2005; Hildebrand et al. 2006). 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, which typically provide an extensive collection of syntactic control structures[3].

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 objects 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 (SIMILE 2005; m.c. schraefel et al. 2005; Hildebrand et al. 2006; Kiryakov et al. 2004; Hyvönen and Mäkelä 2006). A different approach is seen in DBin (Tummarello et al. 2006) and Haystack (Quan and Karger 2004), which allows the user to select predefined queries.

[3] http://lucene.apache.org/java/2_3_2/queryparsersyntax.html

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 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 analysed (Schreiber et al. 2006; Celino et al. 2006; Guha et al. 2003; Davies and Weeks 2004; Ding et al. 2004). 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 (Kiryakov et al. 2004; Mika 2006; Heflin and Hendler 2000; Metaweb 2007). 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. 2007a). Faceted browsing is applied to Semantic Web data by (SIMILE 2005; m.c. schraefel et al. 2005; Hildebrand et al. 2006; Suominen et al. 2007a; Fluit et al. 2003a; Hyvönen et al. 2005; Oren et al. 2006) as well as by the company Siderean in the Seamark Navigator[4].

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 typed in already. Hyvönen and others describe the idea of semantic autocompletion and several implementations in (Hyvönen and Mäkelä 2006). In faceted interfaces autocompletion is often used within a single facet (SIMILE 2005; m.c. schraefel et al. 2005; Hildebrand et al. 2006; Kiryakov et al. 2004).

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

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.

2.3.2 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.

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 (Schreiber et al. 2006; Celino et al. 2006; Hyvönen et al. 2005). For images that occur in web pages the contextual text provide an alternative source (Celino et al. 2006; Group 2006). 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[7], 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 in this graph. Naive approaches to semantic search are computationally too expensive and increase the number of results dramatically. Systems thus need to find a way to reduce the search space and to determine which semantically related objects are really relevant.

Inspired by the semantic continuum described by Ushold (Uschold 2003), we distinguish three levels of semantic matching: graph traversal, explicit use of the-

[5] http://www.openlinksw.com/
[6] http://www.swi-prolog.org/packages/semweb.html
[7] http://lucene.apache.org/

sauri 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. 2005). 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 (W3C 2005) as a standard representation for thesauri, semantic matching with hierarchical broader term (BT) and narrower term (NT) and the associative related term (RT) 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 thesauri 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 service (Binding and Tudhope 2004a), which in turn formed the basis for the experimental SKOS API[8].

RDFS/OWL reasoning can also influence the search results. Several systems support RDFS subsumption once an RDF class is selected in the interface (Lei et al. 2006; Tummarello et al. 2006; Auer et al. 2006; Duke et al. 2007). In Dose, specialisation and generalisation over the subclass hierarchy is 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 2007) these are extensively used to model the identity between extracted entities.

2.3.3 Presentation of Results

We describe how explicit semantics are used to extend the baseline functionality in the presentation of search results, and the techniques that are used to visualise the results in the interface. As a baseline we consider the presentation of search results by popular search engines such as Google: the selected information for presentation is the URI or label of the result, surrogates of the content (e.g. text snippets or

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

image thumbnails) and, optionally, additional information such as the file size. The results are typically organised 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, *organising* the results and *ordering* the results. In addition, we discuss the function of user feedback on the results.

Selecting what to present — This issue is tightly bound to the question what the search engine considers to be a "result". If the result is a Web page or other information object, 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) create templates for specific user tasks and domains. Display vocabularies such as Fresnel (Bizer et al. 2005), as used by Longwell (SIMILE 2005), provide full control over what data to select for presentation and how to present it.

Organising 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 behaviour is found, systems described in (SIMILE 2005; Hildebrand et al. 2006; 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.

Ordering of results — The order of the search results can determined with different techniques. Ranking of results based on relevance is a well covered topic 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 an often used syntactic measure 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 focussed subgraphs (Kleinberg 1999) is used.

User Feedback — In our study, we did not encounter typical IR user feedback where the matching and ranking algorithms is influenced by the user's feedback. We mainly encountered user feedback to disambiguate, specialise, generalise or expand 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 (DERI 2007; Davies and Weeks 2004; Duke et al. 2007; Berlin 2007). Alternatively, queries can be specialised or generalised 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 use 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 organise 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 straight-forward interface that directly reflects the structure of the selected data and how it is organised. 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, visualising the data as a graph may seem a straightforward choice. However, from a user interface perspective, "'big fat graphs" quickly become unmanageable (m.c. schraefel and Karger 2006), with only a few exceptions including the visualisation of social networks between small groups of people (Mika 2005). We discuss some other visualisation techniques (see (Geroimenko and Chen 2003) for a more extensive overview) we encountered.

Tagclouds indicate the importance of textual metadata with variations in the font size. OpenAcademia (Mika 2006) visualises the concepts related to research publications and search. DBpedia.org (Berlin 2007) presents the available RDF types of the search results.

Clustermaps visualise the overlap between classes of instances, without needing an explicit concept representing this overlap (Fluit et al. 2003b). For example in AutoFocus a clustermap visualises the results of individual constraints as well as result sets that satisfy multiple constraints (Fluit et al. 2003a).

Data type-specific visualisations are used in several systems to present space and time on a geographical map, timeline or calendar. The Simile timeline[9], several map visualisation 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. 2006) and Disco (Bizer and Gauß 2007) are based on the notion of a *concise bounded description*or CBD[10] 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[11]Noadster (Rutledge et al. 2005) and E-culture (Schreiber et al. 2006) also include statements where the URI plays the role of the property.

2.4 Discussion

In a survey we investigated the search functionality and result presentation of 35 Semantic Web applications. We conclude that the applications support a wide variety of different types of tasks and provide access to different types of data sets. In addition, they provide different types of support for the three stages of the search process: query formulation, search algorithm and result presentation. It, however, remains unclear how well these technologies improve support for end-users. With a few notable exceptions (Ding et al. 2004; Sure and Iosif 2000), the search algorithms analysed in this study are not or only briefly evaluated on the quality of their search results for end-users. A similar argument applies to the interface components found in our study. For none of the systems we could find user evaluations that would stand the criteria commonly found in the HCI community.

In Information Retrieval, there is a long tradition of evaluating 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 conclude that given a specific domain and search task it is difficult to determine how the semantically-rich graph structure should be used to benefit the end user. In the following three chapters we, therefore, investigate support for specific tasks and use qualitative evaluations to gather insights in the requirements to support end-user access to semantically-rich linked data.

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