

Chapter 7

Semantically Enhanced Search and Browse

Alistair Duke and Jörg Heizmann

Abstract Squirrel, a search and browse tool that provides access to semantically annotated data is described. The tool offers a hybrid approach to search allowing the user to enter simple search terms and then refine their search or browse to related material through the presentation of appropriate metadata. Squirrel builds upon and integrates a number of the semantic technology components described elsewhere in the book. These include machine learning and information extraction components which generate, extract and manage semantic metadata contained within and about textual documents at index time. A number of run-time components have also been integrated to deliver an enhanced user experience such as natural language generation which provides natural language summaries of knowledge held in formal (ontological) structures; device independence which allows the tool to be run on multiple devices; result consolidation which presents the most relevant textual content of result documents rather than a simple list of results; and a natural language interface which translates natural language queries into structured queries formulated with respect to a given ontology.

7.1 Introduction

This chapter describes Squirrel, a search and browse tool that provides access to semantically annotated data. Search is seen as a key application that can benefit from semantic technology with improvements to recall and precision versus conventional Information Retrieval techniques. Squirrel builds upon and integrates a number of the semantic technology components described elsewhere in this book. These include machine learning and information extraction components which generate, extract and manage semantic metadata contained within and about textual documents at index time. A number of run-time components have also been

A. Duke (✉)
Next Generation Web group, BT Research, Ipswich, IP5 3RE, UK
e-mail: alistair.duke@bt.com

integrated to deliver an enhanced user experience such as natural language generation which provides natural language summaries of knowledge held in formal (ontological) structures; device independence which allows the tool to be run on multiple devices; result consolidation which presents the most relevant textual content of result documents rather than a simple list of results; and a natural language interface which translates natural language queries into structured queries formulated with respect to a given ontology.

A hybrid approach has been adopted to provide a balance between the speed and ease of use of simple free text search and the power of semantic search. Users are able to enter simple search terms and then refine their search or browse to related material through the presentation of appropriate metadata. Alternatively, users can express a query using natural language which is analysed and converted into a semantic query.

The chapter is structured as follows. The next section introduces a scenario which illustrates both the requirements and benefits of enhanced search and browse. Section 7.3 describes the salient features of the supporting components that comprise Squirrel and presents an architecture for the system. In Sect. 7.4 further detail is provided about the user experience with a description of the important features of the interface. Section 7.5 discusses related work and outstanding issues whilst we conclude in Sect. 7.6 with a view of the way forward.

7.2 Scenario

Squirrel has been trialled in a case study which is developing an improved Digital Library for BT, as discussed elsewhere in this volume. The following scenario describes a Digital Library user carrying out a knowledge seeking task, making use of the features provided by Squirrel and its supporting components.

The user has an initial high level goal to seek information about the field of Home Health Care and has little idea about what is contained in the library which might be of use. He first enters “Home Health Care” into the search box and hits the “Go!” button. The first result screen includes a summary of the sorts of resources that have been found that might be able to meet his needs. These include the textual resources from the library that is that there are a number of journal articles, conference papers, periodicals and web pages (shared by library users and indexed as library resources) that match his query. In addition, there a number of library topics that also match his query including one which is itself called “Home Health Care”. Further matches include a number of organisations from a domain knowledge base whose description includes the search term.

In addition to this summary, the user is also presented with the top ranked textual resources (in typical search engine style). This simple list of documents is augmented with the ability to refine the search based on the properties of the documents in the result set, including a hierarchical display of the topics of the results documents, date of publish, author name, etc.

Our user decides that they are interested in current affairs related to their topic rather than scientific papers. He chooses to refine the search to the periodicals section of the library. The topic hierarchy is rebuilt to accommodate the documents in the refined results set. Furthermore, our user is interested in the financial aspects of home health care and notices that a topic called “Economic Conditions” has been presented. He then chooses this topic to further refine their results set.

A short taster of each result together with its title and other metadata is presented. This is enhanced by highlighting the named entities that have been recognised in the text such as people, organisation, locations, etc. Our user places their mouse on one such highlighted term, “United Health Products” and is shown a pop-up which tells him that this is a company. Intrigued about the company, he clicks on the term and is taken to a screen giving further information about the company including a natural language summary about the company as well as related entities such as the people who work for it. There is also a link to all of the documents where this company (including any of its aliases such as UHP, etc.) is mentioned. Our user clicks on this link but then decides he would rather view the results as a consolidated summary rather than a list of discrete results. In this view the most relevant portions of the result documents are shown to the user meaning he can quickly view the available material rather than having to navigate to several different pages. The user again refines the contents of the summary by selecting one of more of the topics that have been assigned to the presented material.

Our user is also able to use the natural language interface by entering a natural language query. The addition of a question mark identifies it as such to Squirrel. Thus our user might commence their search by entering the query “Which articles are about ‘Home Health Care’?” followed by “Which periodicals are about Economic Conditions?”

7.3 Architecture

Squirrel integrates a number of components developed in the SEKT project. This section briefly describes the features of the components and then presents an architecture illustrating how they have been integrated.

7.3.1 *Proton*

PROTON is a lightweight general purpose ontology developed in SEKT which includes a module specific to the knowledge management domain. PROTON is used by SEKT components for metadata generation and as a basis for knowledge modelling and integration.

A world knowledge base (originally developed as part of the KIM platform (Popov et al. 2004)) has been expressed in PROTON. The knowledge base is

comprised of more than 2,00,000 entities. These are gathered semi-automatically from a range of high quality public data sources. It includes around 36,000 locations, 1,40,000 companies and other organisations and the world's pre-eminent politicians, businesspeople and technologists.

7.3.2 *Full-Text Index*

The SEKT Integration Platform (SIP) provides a full-text indexing facility which is used to index the various forms of textual resource including the labels and literal properties of the ontological instances and classes. This allows metadata, such as authors' names, topic names, knowledge base entities, to be discovered and presented to the user as a way to refine their search or as an alternative way to find documents, since the metadata is always related to a set of documents in some way. The index of ontological instance will, given a search term, provide a set of matching URIs which can then be used to query the ontology repository to extract further details of the concept they refer to.

7.3.3 *KAON2*

Squirrel uses KAON2 (described in Chap. 2) as an ontology layer, providing access to ontological instances and reasoning support. KAON2's facility to extract ontology instances from relational databases via a mapping function allows a great increase in the number of instances accessed (versus a file based approach where all instances must be stored in memory). KAON2 also supports the Description Logic-safe subset of the Semantic Web Rule Language (SWRL) which allows, through formal reasoning, knowledge to be presented against concepts that goes beyond that provided by the structure of the ontology. For example, one of the attributes displayed in the document presentation is "Organisation". This is not an attribute of a document in the PROTON ontology; however, affiliation is an attribute of the Author concept and has the range "Organisation". As a result, a rule was introduced into the ontology to infer that the organisation responsible for a document is the affiliation of its lead author. Other rules have also been introduced including one allowing the "collaborates with" attribute of an author to be inferred by querying the ontology for all the papers where the author in question has co-authored with others and then inferring that these authors are his or her collaborators.

7.3.4 *Natural Language Generation*

Natural Language Generation (NLG), described in the next chapter, is used by Squirrel to provide natural language summaries on ontological entities. The PROTON world knowledge base contains a rich source of knowledge about entities, which may

be mentioned in the documents indexed. When a Squirrel search is carried out and certain people, organisations or other entities occur in the result set, the user is presented with a summary of information regarding those entities. In addition, document results can be enhanced with brief descriptions of key entities that occur in the text. For example, the knowledge base contains company related metadata, which can be presented to the user as natural language. An example is shown further below in Fig. 7.4.

7.3.5 DIWAF

The SEKT Device Independence Web Application Framework is a server side application, which provides a framework for presenting structured data to the user (Glover and Davies 2005). The framework does not use a mark-up language to annotate the data. Instead, it makes use of templates, which are “filled” with data, rather like a mail merge in a word processor. These templates allow the selection, repetition and rearrangement of data, interspersed with static text. The framework can select different templates according to the target device, which present the same data in different ways.

Squirrel has been built as a DIWAF application. This enables the applications to be easily adapted to alternative device or context requirements (such as different languages). Currently, templates are available to deliver the application to users via a WAP-enabled mobile device, via a PALM PDA and via a standard web browser.

7.3.6 Ontology Generation

Ontology Generation (OG) (see Chap. 2) can automatically identify an ontology from the content of the documents and then classify the documents according to the ontology (Fortuna et al. 2005). In SEKT this process is generally carried out at index-time which allows the knowledge engineer to modify the generated ontology if required. The process results in hierarchical topic ontology and a set of document classifications into that ontology (in accordance with PROTON). These results are used by Squirrel to present the topic hierarchy thus allowing the user to refine their search by topic. The OG component can also be used by Squirrel to generate clusters of documents at query-time. This could be used in the general case where some or all of the documents being searched have not been classified at index-time.

7.3.7 User Profile Construction and Usage

The PROTON ontology includes concepts allowing the interests of users to be stored and modelled as a profile. Such a profile is an important step in providing relevant knowledge to people and is used within Squirrel to personalise the search experience. The profile allows the search tool to predict what the user is interested

in based upon their observed interests. Profiles can be manually or automatically created. An automatic approach places less burden on the user than a manual one but relies on the assumption that the techniques used to collect the profile are accurate (which is why a combination of automatic and manual approaches is often adopted). The approach adopted in SEKT has been to observe the web browsing habits of the individual users and to extract the topics associated with the pages that have been accessed. A level of interest for each of the topics is also determined.

In PROTON, the profile consists of an expression of both long-term and short-term interest in topics from a PROTON topic ontology. The Squirrel search tool makes use of these profiles to modify the way the results are presented to the user, providing context to the user's search query. When ranking documents, the topics of the documents in the result set can be considered against the level of interest in topics in the user's profile. Thus documents with a topic recorded in the profile with a strong short-term interest would be presented first. Topics are related to other topics (either by a sub or super topic or some other weaker relation) so these relationships can also be considered to support the application of profiles.

7.3.8 *Massive Semantic Annotation*

Squirrel makes use of the results of a Massive Semantic Annotation (see Chap. 6) process, carried out at index time. The function uses KIM (Popov et al. 2004), which employs GATE's ontology-based information extraction module to identify named entities in texts. For each of these it carries out instance disambiguation; that is, it either identifies an existing PROTON instance of which the entity is an occurrence or creates a new instance. An annotation (which consists of the instance URI, the document it occurs in and its location within the text) is stored using KIM's highly scalable (due to the very large number of annotations) knowledge store.

In order to support user responsive entity presentation and browsing, Squirrel makes use of the OWLIM semantic repository (Kiryakov et al. 2005). OWLIM is considered to be the fastest OWL repository available. The dual repository approach adopted by Squirrel is justified by the benefits that each repository provides. KAON2 offers relational database mapping and rule support whilst the scalability and speed of OWLIM are suited to handling the volume of data resultant from the semantic annotation process.

7.3.9 *Text Segmentation*

During the indexing process, documents are segmented at topical boundaries in order to support the presentation of consolidated results to the user. This uses a C99 segmenter (Choi 2000) enhanced to consider the distribution of named entities and key phrases identified in the text in addition to the distribution of words. Following

segmentation the subdocuments are classified using the TextGarden classifier described above. These classifications are used in two ways by Squirrel. Firstly, to allow the user to refine their summaries based upon topics of interest and secondly to reorder the presentation of the summary based upon the topics in the user's profile.

7.3.10 *Natural Language Querying*

The ORAKEL (Cimiano et al. 2007) natural language interface translates natural language queries to structured queries formulated with respect to a given ontology. This translation essentially relies on a compositional-semantic interpretation of the question guided by two lexica: a domain-specific and a domain-independent one. As the name suggests, the domain-independent lexicon is encoded into the system a priori and captures the domain-independent meaning of prepositions, determiners, question pronouns etc., which typically remain constant across domains. The domain-specific lexicon needs to be created by a lexicon engineer who is assumed to create the lexicon in various cycles. In fact, ORAKEL builds on an iterative lexicon development cycle in which the lexicon is constantly updated by the lexicon engineer on the basis of the questions the system failed to answer so far. The end users are then able to directly interact with the Digital Library data using natural language questions, which are translated into SPARQL queries. The underlying mechanism, however, is hidden from the users – the only thing users need to do is to input the query just as their normal questions and then they get the result from the portal. The obvious benefit of natural language querying is thus that users can pose fairly complex queries to the system without having to learn a formal language such as SQL or SPARQL. Of course, as for keyword-based retrieval, they will need to get used to the vocabulary understood by the system.

Figure 7.1 gives an overview of the ORAKEL system, which has been designed in a flexible manner allowing to easily replace the system's target query language. As the architecture described here relies on the KAON2 system as inference engine and knowledge repository, ORAKEL was adapted to generate SPARQL queries which are supported by KAON2. Further, a lexicon was generated for the Proton ontology, which specifies the possible lexical representations of the ontology elements in the users' queries. This lexicon was constructed in three iterations: one initial iteration of 6h and 2 follow-up iterations of 30min each in which the questions not answered by the system were examined.

7.3.11 *Architecture*

The components described above have been integrated as shown in the architecture in Fig. 7.2. The figure shows a complete end-to-end architecture for indexing and querying.

The sources of textual data, shown at the bottom of the Fig. 7.2, are stored together with their associated metadata in a database. These sources may be (for example) records of bibliographic data (as in the SEKT Digital Library case

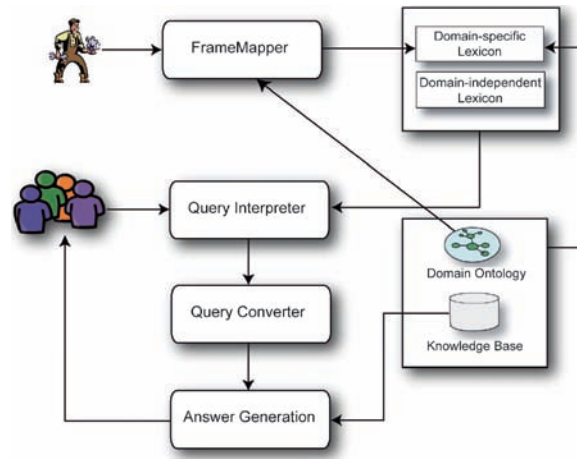


Fig. 7.1 Overview of ORAKEL natural language system

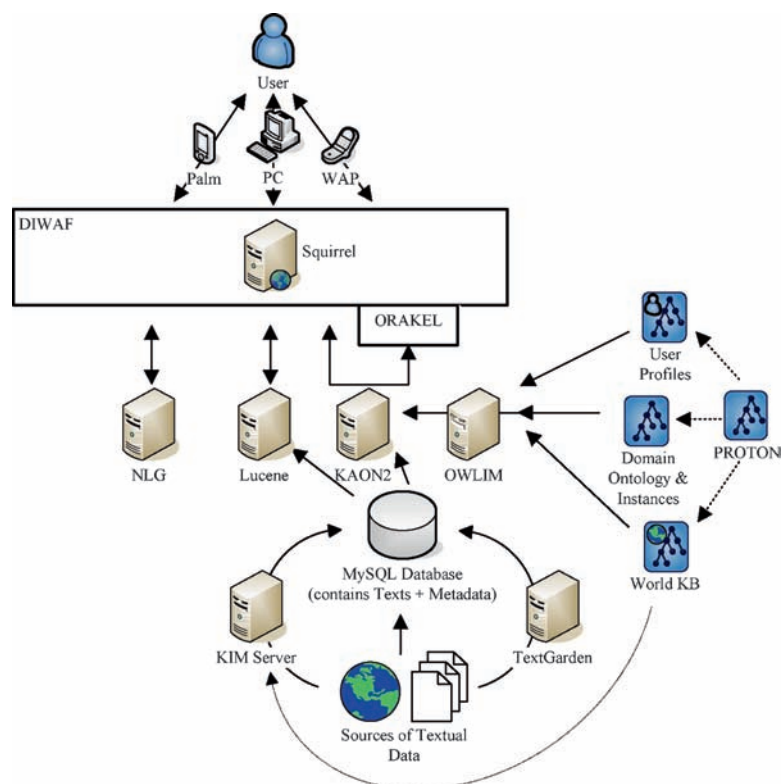


Fig. 7.2 Squirrel architecture

study), webpages identified by a crawler, or legal judgments (as in the SEKT Legal case study). At this stage, the metadata can be augmented by one or more entity extraction or classification components.

Once these steps have been carried out, the contents of the database can then be indexed by SIP as described in Sect. 7.3.2.

A parallel activity to free-text indexing is to load into KAON2 the contents of the database, the PROTON ontology, its world knowledge base and user profiles.

User queries are first passed to SIP in order to retrieve a set of matching documents and entities. Squirrel examines the results of this query and sorts them into entity results and document results. An entity result consists of the URI of the entity. For each of these URIs, Squirrel queries KAON2 via the programming API to determine the type (PROTON class) of the entity and the properties associated with the type for which the entity may have values. Squirrel can then further query KAON2 to determine the values of these properties (if indeed there are any).

Following this, Squirrel is able to present the “meta-result” to the user. This is a summary of the types and number of entities it has found and the types and number of documents (the type of document will typically indicate its source (e.g., webpage, RSS feed, journal article, conference paper). The user is presented with a set of options, which allow them to determine which path they wish to take for their search. The meta-result is accompanied by the default initial result display, which is a list of documents (of any type) that satisfy the query. The first 10 documents from this list are shown, ranked against the query. If a user profile is available, the ranking is adjusted to reflect the topics of interest in the profile. The user interface and the various options provided by the meta-results are described further, with examples, in Sect. 7.4.

When displaying document results, Squirrel queries the database to determine if any annotations are associated with each document. These can then be integrated in the display as highlighted portions of text, which allow the user to “mouse over” to gain further details such as PROTON class and main alias. The user is also given the ability to refine the document set by topic.

When displaying entity results, Squirrel can call the NLG web service (described in Sect. 7.3.4) to generate a summary for particular entities.

The natural language interface supported by ORAKEL is utilised when users append a query with a question mark. As described in Sect. 7.3.10, ORAKEL interacts with the data via KAON2 and its SPARQL interface.

7.4 Interface Description

7.4.1 Initial Search

Users are permitted to enter terms into a text box to commence their search. This initially simplistic approach was chosen based on the fact that users are likely to be comfortable with it due to experience with traditional search engines. If they wish,

the user can specify which type of resource (from a configurable list) they are looking for (e.g., publications, web articles, people, organisations) although the default is to search in all of them.

The first task Squirrel carries out after the user submits a search is to call the SIP index and then use KAON2 to look up further details about the results, which may be textual resources or ontological entities. In addition to instance data, the labels of ontological classes are also indexed. This allows users to discover classes and then discover the corresponding instances and the documents associated with them without knowing the names of any of the instances; for example, a search for “Airline Industry” would match the “Airline” class in PROTON. Selecting this would then allow the user to browse to instances of the class where they can then navigate to the documents where those instances are mentioned. This is an important feature since with no prior knowledge of the domain it would be impossible to find these documents using a traditional search engine.

Textual items can be separated by their type, for example, Web Article, Conference Paper, Book, etc. Squirrel is then able to build the meta-result page based upon the textual content items and ontological instances that have been returned.

7.4.2 *Meta-Result*

The meta-result page is intended to allow the user to quickly focus their search as required and to disambiguate their query if appropriate. The page presents the different types of result that have been found and how many of each type. In order not to introduce unnecessary overhead on the user, the meta-result page also lists a default set of results allowing the user to immediately see those results deemed most relevant to their query by purely statistical means.

The meta-result for the “home health care” query is shown in Fig. 7.3 under the sub-heading “Matches for your query”. The first items in the list are the document classes. Following this is a set of matching topics from the topic ontology. In each case, the number in brackets is the number of documents attributed to each class or topic. Following the topics, a list of other matching entities is shown. The first five



Fig. 7.3 Meta-result

matching entities are also shown allowing the user to click the link to go straight to the entity display page for these. Alternatively they can choose to view the complete list of items. Squirrel can be configured to separate out particular entity types (as is the case with topics and organisations as shown in Fig. 7.3) and display them on their own line in the meta-result.

The returned documents can be ranked according to the user's profile if one has been specified. The algorithm to perform this ranking checks to see which documents are attributed to topics that exist in the profile and then adjusts the ranking using the degree of interest for those topics. For each document, the degree of relevance provided by SIP (which is between 0 and 1) is added to the cumulative degree of interest from the topics in the profile that are also attributed to the document. The documents are then re-ranked based upon these new relevance figures and displayed to the user. The algorithm can be formulated as shown in the following equation.

$$R = r + \sum_{i=1..k} \begin{cases} w(t_i) & t \in U \\ 0 & t \notin U \end{cases} \quad (7.1)$$

Where R is the degree of relevance for a document and r is the (statistical) degree of relevance provided by SIP. A topic attributed to the document is denoted by t_i where i is the index of a topic in the set of attributed topics. $w(t_i)$ denotes the level of interest in the topic t_i if t is a member of the set of topics in the user's profile, U . If the topic is not in the profile then no adjustment is made.

The algorithm attempts to maintain a balance between results with a high relevance ranking mainly due to the SIP result and those that are principally deemed to be of interest according to the profile. Thus, if the user has just switched their focus, the results should not skew too far in favour of the profile which might be slightly biased towards their previous focus.

7.4.3 Refining by Topic

Alongside the results, the user is presented with the topics associated with the documents in the result set. Not all topics are shown here since in the worst case where each document has many distinct topics the list of topics presented to the user would be unwieldy. Instead an algorithm takes the list of topics that are associated with the collection of documents in the result set, and generates a new list of topics representing the narrowest "common ancestors" of the documents' topics.

The algorithm works by repeatedly applying two phases – an "extend" phase, which generates parents of the current topic list, and a "sweep" phase, which merges together identical parents. The algorithm terminates when there is only one topic in the list (a single common ancestor), or when there are no more parents. The set of narrowest "common ancestors" for the whole result set generally provides a more manageable topic list from which the user can refine the results.

Having selected a topic and viewed the subset of documents from the result set, the user can switch to an entity view for the topic. The user can also reach this view by selecting a topic from the meta-result section. Instead of showing the documents of the topic, the meta-data for the topic is shown, which includes the broader, narrower and related topics. This allows the user to browse around the topic ontology. Each topic is shown with the number of documents it contains in brackets after it. Two links to document results are also shown. The first takes the user to a list of documents that have been attributed to the topic itself. The second takes the user to a list of documents that include all subtopics of the current topic. The layout of entity views in Squirrel are defined by templates. This allows the administrator to determine what meta-data is shown and what is not. The use of these templates is discussed further in Sect. 7.4.6 where a “Company” entity view is described.

7.4.4 Attribute-Based Refinement

Each document result list has a link, which opens a refiner window. This allows the user to refine the results based upon the associated metadata. The metadata shown and the manner in which it is displayed are configurable through the use of entity type specific templates that are configured by an administrator or knowledge engineer. Documents can be refined by the user based upon their authors, date of publication, etc. The approach adopted has been to allow the user to enter free text into the attribute boxes and to re-run the query with the additional constraints. An alternative would be to list possible values in the result set. However, the potential size of this list is large and is difficult to present to the user in a manageable way. The disadvantage of the free-text approach is that the user can reduce the result set to zero by introducing an unsatisfiable constraint – which is obviously undesirable. Squirrel attempts to address this by quickly showing the user the number of results as soon as the constraints have been set. The user can then modify them before asking Squirrel to build the result page.

7.4.5 Document View

The user selects a document from the reduced result set, which takes them to a view of the document itself. This shows the meta-data and text associated with the document and also a link to the source page if appropriate – as is the case with web-pages. Since web-pages are published externally with specific formatting data, the text of the page is extracted at index-time. Any semantic mark-up that is applied at this stage can then be shown on the extracted text at query-time. However, the user should always be given the option to navigate to the page in its original format. Figure 7.4 shows a screenshot of the document view.

The document view also shows the whole document abstract, marked-up with entity annotations. “Mousing over” these entities provides the user with further

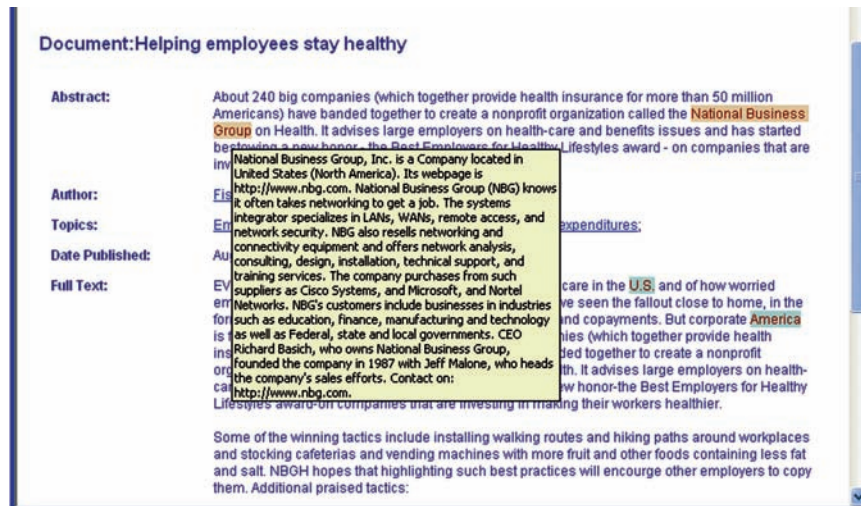


Fig. 7.4 Document view

information about the entity extracted from the ontology using the NLG component. Clicking on the entity itself takes the user to the entity view.

7.4.6 Entity View

The entity view for “Sun Microsystems” is shown in Fig. 7.5. It includes a summary generated by the NLG Web Service described in Sect. 7.3.4. The summary displays information related not only to the entity itself but also information about related entities such as people who hold job roles with the company. This avoids users having to browse around the various entities in the ontology that hold relevant information about the entity in question. The relationship between people and companies is made through a third concept called JobPosition. Users would have to browse through this concept in order to find the name of the person in question.

7.4.7 Consolidated Results

Users can choose to view results as a consolidated summary of the most relevant parts of documents rather than a discrete list of results. This view is appropriate when a user is seeking to gain a wider view of the available material rather than looking for a specific document. The view allows them to read or scan the material without having to navigate to multiple results. Figure 7.6 shows a screenshot of a summary for a query for “Hurricane Katrina”. For each subdocument in the summary the user is able to view the title and source of the parent document, the

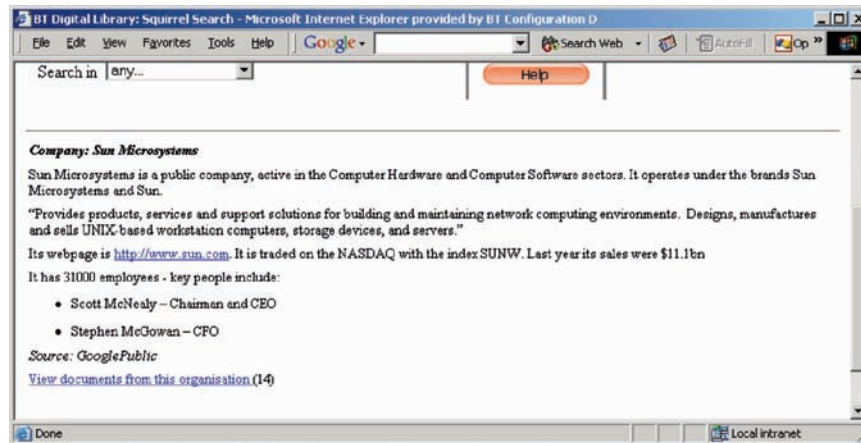


Fig. 7.5 Company entity view

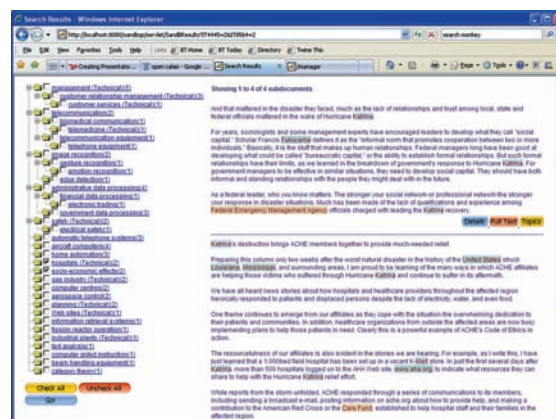


Fig. 7.6 Consolidated results

topics into which the subdocument text has been classified or navigate to the full text of the document. A topic tree is built which includes a checkbox for each topic. These allow the user to refine the summary content by checking or unchecking the appropriate boxes.

7.4.8 Natural Language Interface

The prototype system deployed on the BT Digital Library contained (1) question answering functionality as realised by the ORAKEL system, (2) ontology learning functionality to discover new topics as implemented in the Text2Onto system as well as (3) automatic text classification functionality directly integrated into the KAON2 inference engine as a built-in predicate. The resulting question answering

prototype thus not only allows users to ask questions about existing metadata in the BT Digital Library, but also about topics found by Text2Onto as well as to invoke the automatic text classification system at runtime. The net results are that a variety of queries about authors, topics etc. of documents could be answered successfully against the domain ontology and database. Examples of such questions are:

- What conference papers did Davies write?
- Which conference papers do you know?
- Who wrote which document?
- Who wrote “The future of web services”?
- Who wrote about “Knowledge Management”?
- What article deals with Photography?
- Which journal articles were written by whom?
- What are the topics of “The future of web services”?
- Which conference papers were classified as religion?
- Which articles are about “Intellectual Capital”?
- What articles were written by Lent and Swami?
- Who is the author of a document that talks about which concept?
- Who wrote which articles about what?
- Which documents are about F-Logic and Insurance?

Furthermore, ORAKEL was also modified to process quoted text by matching it against a text index of the metadata in the database using a special purpose predicate match. The question “*What articles are about ‘Intellectual Capital’?*” is translated into the following SPARQL query:

```
SELECT ?w16 WHERE {
  ?w16 rdf:type
    <http://proton.semanticweb.org/2005/04/protonu#Article>
  .
  ?w16 <http://proton.semanticweb.org/2005/04/protont#hasSubject>
    ?x17.
  ?x17 rdfs:label x18.
  match(x18, "Intellectual Capital")
}
```

The application of our prototype enhances the access to BT’s Digital Library showing the integration of data and sources using an inference engine, precise question answering exploitation of newly discovered topics (dynamically) as well as on-the-fly text classification.

7.5 Discussion and Related Work

A number of emerging products and prototypes exist in the Search and Browse space. This section will now briefly consider how the best of these relate to the work described in this chapter.

ISX Corporation’s Concept Object Web (Starz et al. 2005) gathers information from documents using automated and human extraction techniques and then present

this to users as answers to queries rather than leaving the user to read the set of documents in the query response in order to gain the answer. The response to an initial free-text query consists of summaries of documents together with a list of instances that appear in them split into customisable categories. These instances can be selected and added to the initial query in order to refine the results set. The user can also find out more about the instances. In this case they are shown a knowledge object, which is an aggregation of all the semantic information about the entity including links to the source documents and how each fact was obtained. Users can navigate through the knowledge base by selecting the relation values of the current object. The approach is similar in many respects to that adopted by Squirrel. It provides better support for refinements based upon entities but does not allow the user to refine their search based on the topic of documents or browse around a topic ontology in order to find related documents. Additional facilities offered by Squirrel include NLG and tailoring results according to a user profile.

The /facet (Hildebrand et al. 2006) system adopts a navigational model or facet browsing approach where a user can navigate to heterogeneous resources by making selections on the properties of other semantically related types. This benefits users who do not have a clear idea of what they are searching for or know how it is described. Only those attribute values that, if selected, will lead to at least one result being shown are made available. This ensures the user does not take any “blind alleys”. They can also back up by removing previously chosen attributes from their search. One issue with facet browsing is that where there are many possible selections the interface becomes unwieldy. /facet overcomes this by suggesting possible terms to the user as he types in a text box. Again, only keywords that produce actual results are suggested. An impressive feature of /facet is its ability to build facet selection directly from the metadata with no configuration and at query time. Although /facet does not contain many of the features of Squirrel such as named entity recognition, result consolidation or NLG, the navigation approach adopted and its ability to cope with a large number of facets with no configuration indicate how Squirrel could be extended in this fashion.

The Excalibur product from Convera¹ provides a search interface that allows the user to enter queries and then refine or disambiguate them based on topics that the system knows about; for example, a search for “mobile industry” receives the response:

Did you mean: cell phone as a “telecom”, Industries
 mobile as a “ceiling decor”, Industries

Selecting one of these then tailors the results appropriately indicating that documents have been attributed to topics at index time. In addition, once a user has selected a topic they are then offered related topics:

¹<http://www.convera.com/>

Narrow search: consumer electronics, telecom
 Broaden search: LG cell phone, Motorola cell phone, Nokia cell
 phone, Samsung cell phone, Sony cell phone, camera
 phone, clamshell phone, multimedia phone
 Related: cell accessory, sectoral development, by-product

This indicates that there is some sort of topic hierarchy behind the system, but it is not clear that this extends to a full ontology akin to PROTON. Although the system highlights named entities (such as people) in the text, the user is not able to select these and find out more about them, browse to related ontological entities or even refine a search based on the values of properties that particular entities have. Furthermore, the lack of an ontology behind the entities makes it harder to apply rules of the nature described in Sect. 7.3.

7.6 Conclusion and Future Work

We have described Squirrel, a tool for searching and browsing semantically annotated textual resources. The tool provides a hybrid approach to search allowing the user to enter simple search terms and then refine their search or browse to related material through the presentation of appropriate metadata. The tool integrates a number of state-of-the-art components to provide ontology management, named entity recognition on ontology generation and classification. It also includes a set of novel features such as result consolidation, natural language generation, ontology based user profiling, device independence and a natural language query interface.

The tool has been trialled in the BT Digital Library case study. An account of the evaluation is given in Chap. 17.

A number of potential areas for further development include the ability to identify the relationships between entities that are discovered in the query response. For example, a query such as “Java Microsoft” would match a number of different entities including both a topic and a company. The appropriate entities could be chosen based upon a combination of the popularity in the knowledge base (how many documents contain them as annotations), the user’s profile and finally an order of precedence is specified for the domain (e.g., in the digital library, if a topic is identified then this might be chosen over an entity with the same name). In this case the topic Java and the company entity Microsoft might be chosen and as such it might be appropriate to initially show documents from the topic where annotations of the company have been identified. Similarly a query for “Bill Gates Microsoft” would identify the person and company from the knowledge base. Here, as well as showing documents where annotations of both occur, it might be appropriate to show how the two entities are related, which in this case is via the JobPosition instance relating the person to the company.

The display of entity results could also be improved with the use of user profiles. Where a user searches using a term that closely matches two or more entities there is a need for disambiguation (predicting which of the entities the user is searching

for by matching the results against the profile). For example, if a user is searching for documents written by an author called Davies (of which there are many different separate instances) the correct author can be chosen by matching the topics of the documents the individual authors have written against topics in the user's profile.

A second area for development is concerned with adopting a reduced configuration, faceted browsing approach such as that offered by /facet and described in the previous section.

Finally, there is a need for better and optimised integration between ontology management and full-text search. Currently the two are separate components and must be queried separately, either sequentially (as in Squirrel) or in parallel followed by an intersection step. This has inherent performance issues which could be addressed by a combined approach able to take advantage of optimisation techniques. In a similar vein the use of both OWLIM and KAON2 in Squirrel would be improved via the use of a single repository that is able to offer the best features of both these components. The development of the ORAKEL system will aim at making it more robust as well as at providing enhanced answer generation capabilities beyond returning a mere list of tuples. In some cases, answers could be summarised or at least grouped after some criteria.

Acknowledgment The work described in this chapter was developed as part of the SEKT project, under EU funding (IST IP 2003506826). Further project details may be found at <http://www.sekt-project.com/>.

References

- Choi FYY (2000) Advances in domain independent linear text segmentation'. In Proceedings of NAACL, Seattle, USA, April: 26–33.
- Cimiano P, Haase P, Heizmann J (2007) Porting natural language interfaces between domains – A case study with the ORAKEL system. In Proceedings of the International Conference on Intelligent User Interfaces (IUI): 180–189.
- Fortuna B, Grobelnik M, Mladenić D (2005) Semi-automatic Construction of Topic Ontology', Semantics, Web and Mining. Joint International Workshop, EWMF 2005 and KDO 2005, Porto, Portugal, October 3–7.
- Glover T, Davies J (2005) Integrating device independence and user profiles on the web. BT Technology Journal Vol. 23.
- Hildebrand M, van Ossenbruggen J, Hardman L (2006) /facet: A browser for heterogeneous semantic web repositories'. In Proceedings of the 5th International Semantic Web Conference, Athens, USA, November, 2006.
- Kiryakov A, Ognyanov D, Manov D (2005) OWLIM – A pragmatic semantic repository for OWL. In Proceedings of International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE 2005, 20 November, New York City, USA.
- Popov B, Kiryakov A, Ognyanoff D, Manov D, Kirilov A (2004) KIM – A semantic platform for information extraction and retrieval. Journal of Natural Language Engineering, Vol. 10, Issue 3–4, 2004: 375–392.
- Starz J, Kettler B, Haglich P, Losco J, Edwards G, Hoffman M (2005) The concept object web for knowledge management. Proceedings of the 4th International Semantic Web Conference, Galway, Ireland, November, 2005.

Semantic Knowledge Management
Integrating Ontology Management, Knowledge
Discovery, and Human Language Technologies
Davies, J.F.; Grobelnik, M.; Mladenić, D. (Eds.)
2009, X, 252 p., Hardcover
ISBN: 978-3-540-88844-4