9

Semantic Search

John Davies, Alistair Duke and Atanas Kiryakov

9.1 Introduction

This chapter will examine the use of semantic technology in information retrieval. We discuss the prospects for systems which combine semantics-aware information retrieval (IR) techniques to search information resources with the ability to browse and query against semantic annotations of those resources.

An example of semantic annotation is tagging (or marking up) a web page with the identifiers (e.g. URLs) of people, organizations, locations and other entities referred to. These identifiers can be considered as 'semantic features', since they constitute a feature-space where the resources are characterised by means of entities and concepts (instead of just tokens, lemmata, or stems as in traditional IR). This feature-space is usually of reduced dimensionality – there are a number of different strings which can represent a reference to one and the same concept; as for instance, 'UK', 'U.K.' and 'United Kingdom'. Another interesting quality of the semantic features space is that it is (or at least it can be) structured. One can use a database or a knowledge base (KB[1]) which contains structured information about the entities mentioned in the documents. An example would be a KB holding the facts that a person works for a company, which is registered in some city, which in turn is located in a particular country. This allows for retrieval approaches where probabilistic models are combined with structured queries. An example of this would be a search for documents referring to a person called John, who works for a non-government organisation in Bulgaria. A proper answer to such a request requires concrete factual knowledge, which may not be available in the documents where the person is mentioned. The latter means that for a classical IR system it is theoretically impossible to handle such query.

When semantic features are used as an alternative feature-space one gains the benefits typically associated with reduced dimension models (e.g. LSA[2]): better performance on noisy/redundant data and 'associative' retrieval. Typically, this leads to higher precision and lower recall; in the above example, not all documents containing the token 'john' will be extracted. Recall can be improved through query expansion based on the underlying KB.

[1] KB is a term with various interpretations; we use it as something broader than ontology; see Section 9.1.2.
[2] Latent Semantic Analysis (LSA) is presented in (Landauer and Dumais 1997); it is also discussed in Section 3.3 of Chapter 11 from the perspective of cross-language IR.

Information Retrieval: Searching in the 21st Century edited by A. Göker & J. Davies

Another approach is to use the semantic features not as an alternative, but rather as an extension of the standard feature set used for full-text search. In such cases, semantic search can be characterised as 'low threshold, high ceiling' in the sense that where semantic annotations exist they are exploited for an improved information-seeking experience, but where they do not exist, a search capability is still available. Searching the full text of documents can ensure the high recall desirable in early stages of the information seeking process. In later stages of the search, when the user may typically be more interested in the precision of the retrieval results, it can be advantageous to put more emphasis on searching based on the semantic annotations.

The main objectives and the structure of this chapter can be summarized as follows:

- To outline some limitations of current search technology (Section 9.1.1);
- To introduce the notion of ontologies (Section 9.1.2);
- To introduce the Semantic Web and its relation to semantic search (Section 9.2);
- To discuss markup and annotation of text (Section 9.3);
- To discuss the role of semantic annotations (Section 9.4) and present a specific schema for semantic annotation of texts with respect to named entities (Section 9.5);
- To elaborate how search based on semantic annotations can be defined in IR terms (information need and satisfaction) and implemented (Section 9.6);
- To acquaint the reader with some of the most prominent semantic annotation and search systems (Section 9.7).

9.1.1 Limitations of current search technology

In general, when specifying a search, users enter a small number of terms in the query. The query describes the information need, and is commonly based on the words that people expect to occur in the types of document they seek. This gives rise to a fundamental problem, known as 'index term *synonymy*': not all documents will use the same words to refer to the same concept. Therefore, not all the documents that discuss the concept will be retrieved by a simple keyword-based search. Furthermore, query terms may of course have multiple meanings; this problem can be called 'query term *polysemy*'. As conventional search engines cannot interpret the sense of the user's search, the ambiguity of the query leads to the retrieval of irrelevant information.

Technically, the above two problems can be explained as follows: search engines that match query terms against a keyword-based index will fail to match relevant information when the keywords used in the query are different from those used in the index, despite having the same meaning. This problem can be overcome to some extent through thesaurus-based expansion of the query; this approach increases the level of document recall, but it may result in significant precision decay, i.e. the search engine returning too many results for the user to be able to process realistically.

Users can partly overcome query ambiguity by careful choice of additional query terms. However, there is evidence to suggest that many people may not be prepared to do this. For example, an analysis of the transaction logs of the Excite WWW search engine (Jansen *et al*. 2000) showed that web search engine queries contain on average 2.2 terms. Comparable user behaviour can also be observed on corporate intranets: an analysis of the queries submitted to the intranet search engine of BT[3] over a 4-month period between January and May 2004 showed an average query length of only 1.8 terms.

In addition to difficulties in handling synonymy and polysemy, conventional search engines are of course unaware of any other semantic links between terms (or, more precisely, the concepts which the terms represent). A major limitation of non-semantic IR approaches is that they cannot handle queries which either require knowledge and data which are not available in the documents; or require extraction, explicit structuring, and reasoning about some data. Consider for example, the following query:

'telecom company' Europe 'John Smith' director

[3] http://www.bt.com/

The information need appears to be for documents concerning a telecom company in Europe, a person called John Smith, and a board appointment. Note, however, that a document containing the following sentence would not be returned using conventional search techniques:

> At its meeting on the 10th of May, the board of London-based O2 appointed John Smith as CTO.

In order to be able to return this document, the search engine would need to be aware of the following semantic relations:

- O2 is a mobile operator, which is a kind of telecom company;
- London is located in the UK, which is a part of Europe;
- A CTO is a kind of director.

These are precisely the kinds of relations which can be represented and reasoned over using semantic web technology.

9.1.2 Ontologies

Formal knowledge representation (KR) is about building models[4] of the world (of a particular state of affairs, situation, domain or problem), which allow for automatic reasoning and interpretation. Such formal models are called *ontologies*, whenever they (are intended to) represent a shared conceptualisation (e.g. a basic theory, a schema, or a classification). Ontologies can be used to provide formal semantics (i.e. machine-interpretable meaning) to any sort of information: databases, catalogues, documents, web pages, etc. Ontologies can be used as semantic frameworks: the association of information with ontologies makes such information much more amenable to machine processing and interpretation. This is because formal ontologies are represented in logical formalisms, such as OWL (Dean *et al*. 2004), which allow automatic inferencing over them and over datasets aligned to them. An important role of ontologies is to serve as schemata or 'intelligent' views over information resources[5]. Thus they can be used for indexing, querying, and reference purposes over non-ontological datasets and systems, such as databases, document and catalogue management systems. Because ontological languages are formal logical languages, ontologies allow inference of facts which are not explicitly stated from the explicit data. In this way, they can improve the interoperability and the efficiency of the usage of arbitrary datasets.

An ontology can be characterized as comprising a 4-tuple[6]:

$$O =< C, R, I, A >$$

where C is a set of **classes** representing *concepts* we wish to reason about in the given domain (invoices, payments, products, prices, ...); R is a set of **relations** (also referred to as **properties** or **predicates**) holding between (instances of) those classes (Product *hasPrice* Price); I is a set of **instances**, where each instance can be an instance of one or more classes and can be linked to other instances or to **literal** values (strings, numbers, ...) by relations (*product23* compatibleWith product348; *product23* hasPrice €170); A is a set of **axioms** (if a product has a price greater than €200, then shipping is free).

[4] The typical modelling paradigm is mathematical logic, but there are also other approaches, rooted in information and library science. KR is a very broad term; here we only refer to one of its main streams.

[5] Comments in the same spirit are provided in (Gruber 1992) also. This is also the role of ontologies on the semantic web.

[6] By tuple is meant an ordered list. A more formal and extensive mathematical definition of an ontology is given, for example, in (Ehrig *et al.* 2005). The characterisation offered here is suitable for the purposes of our discussion, however.

The ontologies can be classified as *lightweight* or *heavyweight* according to the complexity of the KR language used. Lightweight ontologies allow for more efficient and scalable reasoning, but do not possess the high predictive (or restrictive) power of the full-bodied concept definitions of heavyweight ontologies. The ontologies can be further differentiated according to the sort of conceptualisation that they formalise: *upper-level* ontologies model general knowledge, while *domain-* and *application-ontologies* represent knowledge about a specific domain (e.g. medicine or sport) or a type of applications (e.g. knowledge management systems). Basic definitions regarding ontologies can be found in (Gruber 1992; 1993) and (Guarino and Giaretta 1995; Guarino 1998).

Finally, ontologies can be distinguished according to the sort of semantics being modelled and their intended usage. The major categories from this perspective are:

- *Schema-ontologies*: ontologies which are close in purpose and nature to database and object-oriented schemata. They define classes of objects, their appropriate attributes and relationships to objects of other classes. A typical usage of such an ontology is that large sets of instances of the classes are defined and managed. Intuitively, a class in a schema ontology corresponds to a table in an RDBMS (relational database management system); a relation – to a column; an instance – to a row in the table for the corresponding class.
- *Topic-ontologies*: taxonomies which define hierarchies of topics, subjects, categories, or designators. These have a wide range of applications related to classification of different things (entities, information resources, files, web pages, etc.) The most popular examples are library classification systems and taxonomies, which are widely used in the KM field. Yahoo and DMoz[7] are popular large scale incarnations of this approach in the context of the web. A number of the most popular taxonomies are listed as encoding schemata in Dublin Core (DCMI 2005).
- *Lexical ontologies*: lexicons with formal semantics, which define lexical concepts[8], word-senses and terms. These can be considered as semantic thesauruses or dictionaries. The concepts defined in such ontologies are not instantiated, rather they are directly used as reference, e.g. for annotation of the corresponding terms in text. WordNet is the most popular general purpose (i.e. upper-level) lexical ontology; it is discussed in greater detail in Section 2 of Chapter 10.

This chapter is mostly concentrated on annotation, indexing and retrieval with respect to schema-ontologies and KBs built with respect to them. The usage of taxonomies for document categorization is discussed in Chapter 8. The usage of lexical ontologies for IR is discussed in Chapter 10.

PROTON (Terziev *et al*. 2004) is a light-weight upper-level schema-ontology developed in the scope of the SEKT project (Davies *et al*. 2005). It is used in the KIM system for semantic annotation, indexing and retrieval. We will also use it for ontology-related examples within this section. PROTON is encoded in OWL Lite and defines about 300 classes and 100 properties, providing good coverage of named entity types and concrete domains (i.e. modelling of concepts such as people, organizations, locations, numbers, dates, addresses, etc.) A snapshot of the PROTON class hierarchy is given in Figure 9.1.

9.1.3 Knowledge bases and semantic repositories

Knowledge base (KB) is a broader term than ontology. Similarly to an ontology, a KB is represented in a KR formalism, which allows automatic inference. It could include multiple axioms, definitions, rules, facts, statements, and any other primitives. In contrast to ontologies, however, KBs are not intended to represent a shared or consensual conceptualisation. Thus, ontologies are a specific sort of KB. Many KBs can be split into ontology and instance data parts, in a way analogous to the splitting of schemata and concrete data in databases. A broader discussion on the different terms related to ontology and semantics can be found in section 3 of (Kiryakov 2006).

[7] http://www.yahoo.com and http://www.dmoz.org respectively.

[8] We use 'lexical concept' here as some kind of a formal representation of the meaning of a word or a phrase. In Wordnet, for example, lexical concepts are modelled as synsets (synonym sets), while word-sense is the relation between a word and a synset.

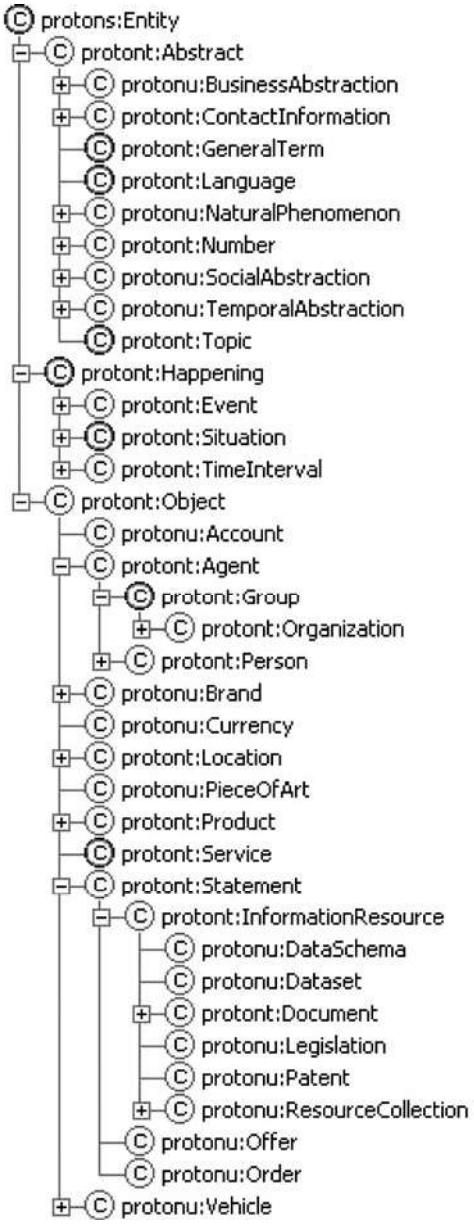

Figure 9.1 A view of the top part of the PROTON class hierarchy

Semantic repositories[9] allow for storage, querying, and management of structured data with respect to formal semantics; in other words, they provide KB management infrastructure. Semantic repositories can serve as a replacement for database management systems (DBMS), offering easier integration of diverse data and more analytical power. In a nutshell, a semantic repository can dynamically interpret

[9] 'Semantic repository' is not a well-established term. A more elaborate introduction can be found at http://www.ontotext.com/inference/semantic_repository.html.

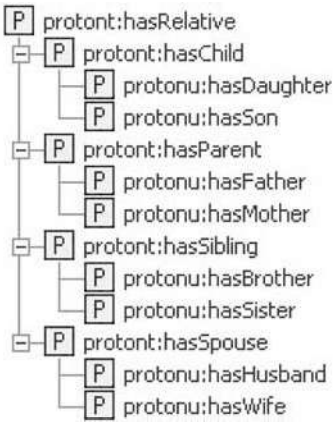

Figure 9.2 A hierarchy of family relationships

metadata schemata and ontologies, which determine the structure and the semantics of data and of queries against that data.

Compared with the approach taken in relational DBMSs, this allows for: (i) easier changes to and combinations of data schemata; and (ii) automated interpretation of the data. As an example, let us imagine a typical database populated with the information that John is a son of Mary. It will be able to 'answer' just a couple of questions: *Who are the son(s) of Mary?* and *Of whom is John the son?* Given simple family-relationships ontology (as the one in PROTON, see Figure 9.2), a semantic repository could handle a much bigger set of questions. It will be able infer the more general fact that John is a child of Mary (because hasSon is a sub-property of hasChild) and, even more generally, that Mary and John are relatives (which is true in both directions, because hasRelative is defined to be symmetric in the ontology). Further, if it is known that Mary is a woman, a semantic repository will infer that Mary is the mother of John, which is a more specific inverse relation. Although simple for a human to infer, the above facts would remain unknown to a typical DBMS and indeed to any other information system, for which the model of the world is limited to data-structures of strings and numbers with no automatically interpretable semantics.

9.2 Semantic Web

Semantic Web is about adding formal semantics to the content on the WWW (and private web-based systems such as company intranets). The current web contains mostly HTML, meant for human interpretation; the machine-processable metadata there is related to the formatting and the layout of the text, but not to its meaning. Berners-Lee (1999) introduced the notion of the Semantic Web thus: 'If HTML and the Web made all the online documents look like one huge **book**, RDF, schema, and inference languages will make all the data in the world look like one huge **database**'. The two major design rationales of the Semantic Web are outlined at the web site of W3C[10] as follows:

- Common formats for the interchange of data;
- Language for recording how the data relates to real-world objects.

As an Internet standardization authority, W3C promotes Semantic Web as one of its key initiatives.

[10] http://www.w3.org/2001/sw/

9.2.1 Semantic web and semantic search

One of the anticipated benefits of the Semantic Web is that information held on it will be accessible through semantic search engines and that such search engines will offer a more effective search capability than that offered by today's keyword-based search engines. Thus, the relationship between Semantic Web and semantic search is two-fold[11]:

- Semantic Web standards and technologies can be used to enable Semantic Search;
- Semantic Search can be the killer application of the Semantic Web.

One of the open issues for the Semantic Web is the establishment of effective information access methods.[12] For instance, classical IR has a single, basic, well-established and understood way of defining and satisfying the information need[13]: it is defined as a set of words (tokens) of interest and is satisfied with a list of documents relevant to those words. As regards the domain of relational databases, the information need is defined as an SQL (Structured Query Language) query and satisfied, typically, through a result table. An obvious question about the Semantic Web is:

How is the information need defined and satisfied within the Semantic Web?

The above question is stated incorrectly – like the current web, the Semantic Web cannot be expected to have a single access method, and therefore a single approach for the definition of the information need. The following questions thus seem more relevant:

- How does the Semantic Web extend the existing information access methods?
- What new information access methods become feasible?

Proposals which answer the above question should unite the following elements: proven user needs, a sound scientific theory, and a robust technology, which can implement efficient applications based on the theory. As outlined in Section 9.1.1, the models employed by the classical IR engines are far from perfect: the information need is poorly defined and imprecisely satisfied. However, search engines are popular because they meet a number of conditions that are critical for a wide acceptance of an information access method in the web context:

(i) To significantly improve the efficiency and effectiveness of users to access information on the web;
(ii) To ensure that no additional skills, effort, discipline, good will, or correctness are required from information providers;
(iii) To offer predictable behaviour and performance.

9.2.2 Basic semantic web standards: RDF(S) and OWL

A family of markup and KR standards were developed, under W3C-driven community processes, as a basis for the Semantic Web. RDF (Klyne and Carroll 2004) is a metadata representation language, which serves as a basic data-model for the Semantic Web. It allows resources to be described through relationships to other resources and literals. The resources are defined through unified resource identifiers (URIs) (, as in XML; e.g. URL). The notion of resource is virtually unrestricted; anything can

[11] Though note that the two are not interdependent: for instance, ontology-based semantic search can be used in knowledge management systems which do not use Semantic Web standards or technology.

[12] An information access method could be considered as a paradigm for: (i) definition of an information need (the question); and (ii) the way that need is satisfied (the answer/result type). This notion is related to the one of "models", from Section 1.2 of Chapter 1.

[13] Of course, this statement reflects an oversimplified view on what IR really does, but still this is the level of understanding that most users have, and it allows them to make use of the technology.

be considered as a resource and described in RDF: from a web page or a picture published on a web to concrete entities in the real world (e.g. people, organisations) or abstract notions (e.g. the number π and the musical genre Jazz). Literals (again as in XML) are any concrete data values e.g. strings, dates, numbers, etc. The main modelling block in RDF is the statement – a triple <Subject, Predicate, Object>, where:

- Subject is the resource, which is being described;
- Predicate is a resource, which determines the type of the relationship;
- Object is a resource or a literal, which represents the 'value' of the attribute.

A set of RDF triples can be seen as a graph, where resources and literals are nodes and each statement is represented by a labelled arc (the Predicate or relation), directed from the Subject to the Object. So-called blank nodes can also appear in the graph, representing unique anonymous resources, used as auxiliary nodes. A sample graph, which describes a web page, created by a person called Adam, can be seen in Figure 9.3.

Resources can belong to (formally, be *instances* of) classes – this can be expressed as a statement through the rdf:type system property as follows: <resource, rdf:type, class>. Two of the system classes in RDF are rdfs:Class and rdf:Property. The instances of rdf:Class are resources which represent classes, i.e. those resources which can have other resources as instances. The instances of rdf:Property are resources which can be used as predicates (relations) in triple statements.

The most popular format for encoding RDF is its XML syntax, (Becket 2004). However, RDF can also be encoded in a variety of other syntaxes. The main difference between XML and RDF is that the underlying model of XML is a tree of nested elements, which is rather different from the graph of resources and literals in RDF.

RDF Schema (RDFS), (Brickley and Guha 2000), is a schema language, which allows for definition of new classes and properties. OWL, (Dean *et al*. 2004), is an ontology language, which extends RDF(S)[14] with means for more comprehensive ontology definitions. OWL has three dialects: OWL-Lite, OWL-DL, and OWL-Full. Owl-Lite is the least expressive of these dialects but the most amenable to efficient reasoning. Conversely, OWL-Full provides maximal expressivity but is undecidable[15]. OWL-DL can be seen as a decidable sub-language inspired by the so-called description logics.

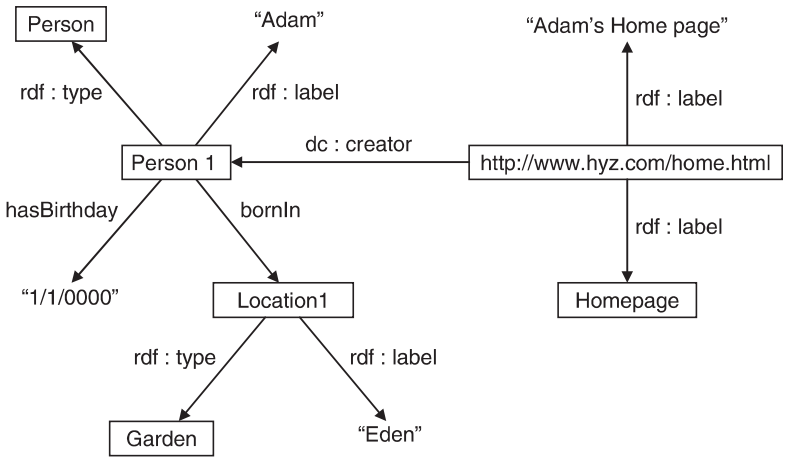

Figure 9.3 RDF graph describing Adam and his home page

[14] RDF(S) is a short name for the combination of RDF and RDFS.

[15] An undecidable logical language is one for which it is a theoretical impossibility to build a reasoner which can prove all the valid inferences from any theory expressed in that language.

These dialects are nested such that every OWL-Lite ontology is a legal OWL-DL ontology and every OWL-DL ontology is a legal OWL-Full ontology.

Below we briefly present all the modelling primitives used in the PROTON ontology, comprising most of the RDFS constructs and few of the simplest ones from OWL:

- All resources, including classes and properties, may have titles (literals[16], linked through property rdf:label) and descriptions or glosses (literals linked through rdf:comment);
- Classes can be defined as sub-classes, i.e. specializations, of other classes (via rdf:subClassOf). This means that all instances of the class are also instances of its super class. For example, in PROTON City is a sub-class of Location.
- In OWL, properties are distinguished into object- and data-properties (instances respectively of owl:ObjectProperty and owl:DataProperty). The object-properties are binary relationships, relating entities to other entities. The data-properties can be considered as attributes – they relate entities to literals.
- Domains and ranges of properties can be defined. A domain (rdfs:domain) specifies the classes of entities to which this property is applicable. A range (rdfs:range) specifies the classes of entities (for object-properties) or data-types of the literal values (in case of data-properties), which can serve as objects in statements predicated by this property. For instance, the property hasSister might typically have the class Person as its domain and Woman as its range. Whenever multiple classes are provided as domain or range for a single property, the intersection of those classes is used.
- Properties can be defined as sub-properties, i.e. specializations of other properties (via rdf:subPropertyOf). Imagine that there are two properties, p1 and p2, for which <p1,subPropertyOf,p2>. The formal meaning of this statement is that for all pairs for which p1 takes place, i.e. <x,p1,y>, p2 also takes place, i.e. <x,p2,y> is also true. The hierarchy of family relationships discussed above (see Figure 9.2) provides a number of intuitive examples of sub-properties.
- Properties can be defined as a symmetric (via owl:SymmtericProperty) and transitive (via owl:TransitiveProperty) ones. If p1 is a symmetric property then whenever <x,p1,y> is true, <y,p1,x> is also true. If p2 is a transitive property and <x,p2,y> and <y,p2,z> are true, it can be concluded that <x,p2,z> is also true. hasRelative is an example of a property which is both symmetric and transitive.
- Object-properties can be defined to be inverse to each other (via owl:inverseOf). This means that if <p1,inverseOf,p2> then, whenever <x,p1,y> holds, <y,p2,x> can be inferred and vice versa. An obvious example is <hasChild, owl:inverseOf, hasParent>.

9.3 Metadata and Annotations

Metadata is a term of wide and sometimes controversial or misleading usage. From its etymology, metadata is 'data about data'. Thus, metadata is a role that certain (pieces of) data could play with respect to other data. Such an example could be a particular specification of the author of a document, provided independently from the content of the document, say, according to a standard like Dublin Core (DC), (DCMI 2005). RDF has been introduced as a simple language that is to be used for the assignment of semantic descriptions to information resources on the web. Therefore an RDF description of a web page represents metadata. However, an RDF description of a person, independent from any particular documents (e.g., as a part of an RDF(S)-encoded dataset), is not metadata – this is data about a person, not about other data. In this case, RDF(S) is used as a KR language. Finally, the RDFS definition of the class Person, will typically be part of an ontology, which can be used to structure datasets and metadata, but which is again not a piece of metadata itself.

A term which is often used as a synonym for metadata, particularly in the natural language processing (NLP) community, is *annotation*. In this section, we discuss annotation of documents in

[16] Recall that literals are values such as strings or numbers.

general, while the next section presents a discussion of 'semantic annotation' in the Semantic Web context.

Annotations on text documents can be distinguished into two groups according to their scope:

- *Document-level annotations*, which refer to the whole document. Such examples are the DC elements (Title, Subject, Creator, etc.);
- *Character-level annotations*, which refer to a fragment of a document, determined by start and end characters. An example might be a comment attached to a particular part of a document. Character-level annotations are usually meant when the term 'annotation' is used for text documents without further clarification.

It is worth mentioning that *hyperlinks* can be considered as a specific sort of character-level annotation, when the metadata is, essentially, a reference to another document or part of document.

Further, annotations can also be distinguished with respect to the way in which they are attached to the text. The basic choices here are:

- *Embedded markup*, when the annotations are incorporated within the document. In this case the metadata is bundled together with the data. Examples are markup languages such as HTML, where the annotations are specified through pairs of start and end tags, e.g. abc<tag>de</tag>gf. When document-level annotations have to be represented this way, they are sometimes attached in a special section at the start or end of the document – one example is the <head> section in the HTML files. An example of character-level embedded annotations is a footnote.
- *Standoff references*, when the annotations are maintained separately from the document to which they refer. In the case of character-level annotations, the reference should also specify the specific part of the document. One approach for this is based on position, e.g. offset and length; another possibility is the usage of some sort of anchoring and linking mechanism. An example of specification based on standoff annotations is TIPSTER (Grishman 1997), a US government-funded effort to advance the state of the art in text handling technologies.

The different types of annotation are shown diagrammatically in Figure 9.4. In his thesis on architectures for language engineering Cunningham (2000), provides an overview of various annotation models and discusses their advantages and disadvantages in the context of text processing systems and applications. Similar analysis, but in the context of open hypermedia systems (OHS), can be found in (van Ossenbruggen *et al*. 2002). Here we will only briefly mention few of the main characteristics of the embedded and the standoff models:

- Embedded markup is not applicable in cases when the author of the metadata has no write-permission to the document;
- Standoff annotations may become inconsistent in the event of change to the document to which they refer;

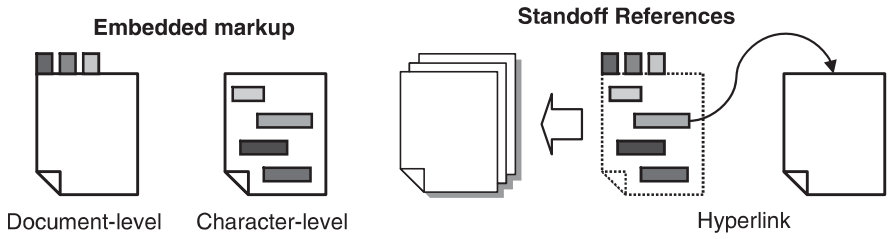

Figure 9.4 Types of Annotations

- Access to embedded annotations requires processing (e.g. parsing) of the documents. Thus, such annotations are not appropriate for applications where random (non-sequential) access to the annotations is important. Conversely, standoff annotations can be maintained and queried efficiently in structured form (e.g. in a database);
- Embedded annotations are simpler to encode, read and manage when the volume of the markup is relatively small. However, they are inappropriate when the volume of the markup becomes comparable to or bigger than that of the text itself;
- Tagging mechanisms based on embedded annotations have difficulties in handling overlapping (as opposed to nested) annotations;
- Embedded annotations should always be distributed together with the document, which can cause IPR issues, unnecessary redundancy or conflict when multiple sets of annotations are available for one and the same document.

9.4 Semantic Annotations: the Fibres of the Semantic Web

If we abstract the current web away from the transport, content type, and content formatting aspects, it could be regarded as a set of documents with some limited metadata, attached to them (document-level annotations about title, keywords, etc.), and with hyperlinks between the documents (see the left-hand side part of Figure 9.5).

What does the Semantic Web add to this picture? Essentially, *semantic* metadata of different kinds, both on the document- and the character-level. Figure 9.5 compares links on the current web to those on the semantic web. Typically, the Semantic Web has a greater number of more meaningful annotations, as compared with the current WWW. Many of those annotations represent links to external knowledge, which constitute a new sort of connectivity that is not presented on Figure 9.5, but is extensively discussed in this section (Figures 9.6 and 9.7).

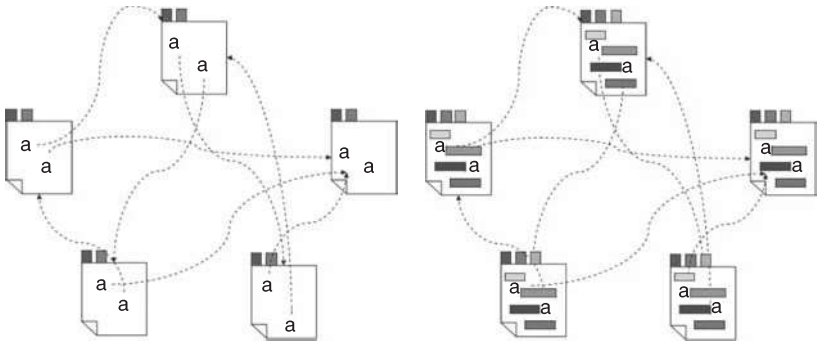

Figure 9.5 The Current WWW (left) and the Semantic Web (right)

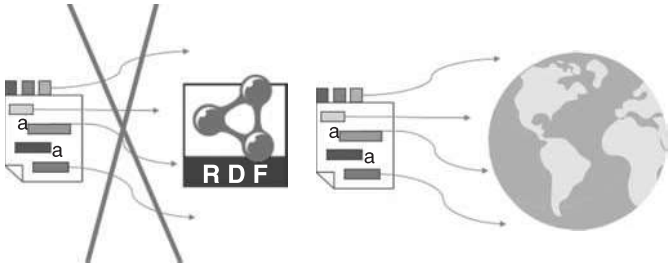

Figure 9.6 Metadata about the world, not about RDF

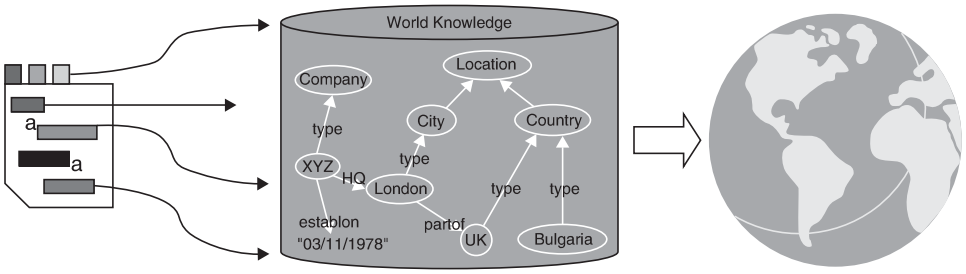

Figure 9.7 Metadata Referring to World Knowledge

In order to uncover the added value of the Semantic Web, it is crucial to elaborate a little further regarding the nature of semantic metadata. Suppose, we add a tag <2134> to some portion of a document as follows '... Abc <2134>xyz...'. Is this metadata useful? Can we call it Semantic? Without further assumptions, the answers are negative. In order to have metadata useful in a Semantic Web context, it should mean something, i.e. the symbols (or expressions or references) that constitute it should allow further interpretation. Interpretation in this context means allowing the assigning of some additional information to the symbols. It is important to realize that interpretation is only possible with respect to something; to some domain, model, context, (possible) world. This is the domain that (the interpretations of) the symbols are 'about'. Obviously, annotations in RDF(S), OWL, or some other language refer to a model of the world. Annotations can be expressed in RDF(S), but they are not *about* RDF(S), as depicted in Figure 9.6.

Further, the metadata can hardly refer to (or be interpreted directly with respect to) the world. Such references cannot be formal and unambiguous. What the semantic metadata can be expected to refer to directly is a knowledge base (KB), a formal model of (some aspect of) the world, as depicted in Figure 9.7. Such a KB specifies some world knowledge which serves as a semantic link from the metadata to the world. Note, that in the Semantic Web context such a KB can be as scattered and heterogeneous as the metadata is. Guha and McCool (2003), consider this KB itself to be the Semantic Web: 'the Semantic Web is not a web of documents, but a web of relations between resources, denoting real world objects'. In our view the Semantic Web is the combination of the KB and the semantic annotations referring to it (not just the KB).

For automatic processing, the interpretations of metadata should be performed automatically by machines in a strict and predictable fashion. This requires a formal definition of how the metadata should be interpreted and, because of this, a formal definition of the context. Assuming that one and the same context can be modelled in different ways, allowing different (and potentially ambiguous) interpretations, what has to be specified is the conceptualisation – as defined in (Gruber 1993): 'a conceptualisation is an abstract, simplified view of the world that we wish to represent for some purpose'. This is where ontologies are used to act as logical theories for the 'formal specification of a conceptualisation' (again in Gruber 1993, see also Section 9.1.2.)

Although the above discussion is informally presented, we consider it rather important for the realisation of the Semantic Web. It is the intuition of the authors that the KR and modelling issues related to the development or generation of useful semantic annotations require more specific attention. RDF(S) and OWL are designed to serve well for data modelling in as diverse and heterogeneous environments as possible. Thus, they provide very little modelling guidance and constraints. For instance, an RDF(S) annotation of an HTML page can include at the same time a definition of the class Person (which is a piece of ontological knowledge), the description of a specific person Mr X (which should normally be world knowledge, part of a KB) and the fact that this person is an author of the web page (which is the only piece of actual metadata describing the web page). We believe that the development of real world Semantic Web applications require some concrete knowledge modelling commitments to be made and the corresponding design and representation principles to be set out. Sections 9.5 and 9.6 below present some specific modelling patterns and applications based on them.

Linguistic approaches for automatic generation of semantic annotations are discussed in Chapter 10 of this volume.

9.5 Semantic Annotation of Named Entities

Semantic annotation of named entities (SANE) is a specific metadata generation process, aiming to enable new information access methods and to extend some of the existing ones. The annotation schema, discussed here, is based on the intuition that named entity (NE, see Section 9.5.1) references constitute an important part of the semantics of the documents in which they occur. Moreover, via the use of KBs of external background knowledge, those entities can be related to formal descriptions of themselves and related entities and thus provide more semantics and connectivity to the web.

In a nutshell, SANE is character-level annotation of mentions of entities in the text with references to their semantic descriptions (as presented in Figure 9.8). This sort of metadata provides both class-level and instance-level information about the entities. Such semantic annotations enable many new types of applications: highlighting, indexing and retrieval, categorization, generation of more advanced metadata, smooth traversal between unstructured text and available relevant structured knowledge. Semantic annotation is applicable for any sort of text – web pages, non-web documents, text fields in databases, etc. Further knowledge acquisition can be performed on the basis of the extraction of more complex dependencies – analysis of relationships between entities, event and situation descriptions, etc.

To use SANE in information retrieval, two basic tasks need to be addressed:

1. Identify and mark references to named entities in textual (parts of) documents and link these references to descriptions of the entities in a KB[17];
2. Index and retrieve documents with respect to the entities they refer to.

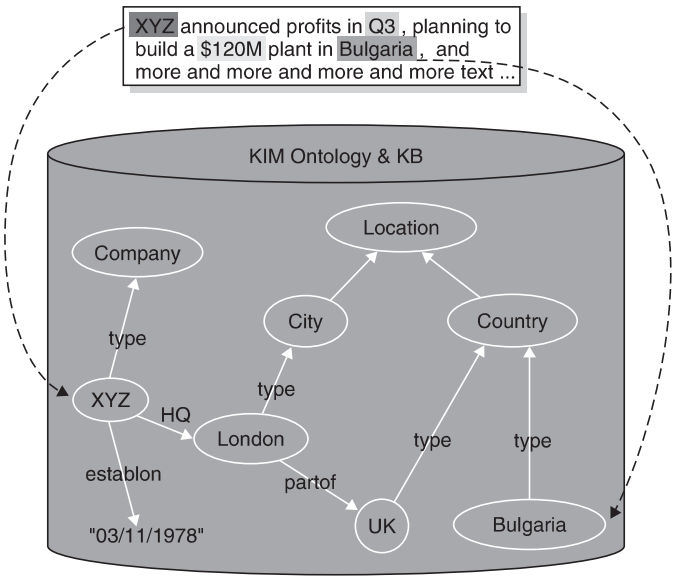

Figure 9.8 Semantic Annotation

[17] There can also be references to various ontologies. On one hand, there could be a direct reference from the annotation to the class of the entity, as it is defined in an ontology. On the other, some instances can be part of ontologies.

The first task resembles at the same time a basic press-clipping exercise, a typical IE[18] task, and hyperlinking. The resulting annotations then provide the semantic data for document enrichment and presentation, which can be further used to enhance information retrieval (the second task) as discussed in Section 9.6.

9.5.1 Named entities

In the Natural Language Processing (NLP) field, and particularly the Information Extraction (IE) tradition, **named entities** (NE) are considered: *people, organizations, locations*, and others, referred to by name, (Chinchor and Robinson 1998). In a wider interpretation, these can also include scalar values (*numbers, dates, amounts of money), addresses*, etc.

NEs should be handled in a different, special way because of their different nature and semantics compared with general words (terms, phrases, etc.). While NEs denote particulars (individuals or instances), the general terms can denote universals (concepts, classes, relations, attributes). Even a basic level of formal semantic definition of general word senses involves modelling of lexical semantics and common sense[19]. On the other hand, useful descriptions of named entities can be modelled on the basis of much simpler and more specific 'factual' world knowledge.

9.5.2 Semantic annotation model and representation

In this section we discuss the structure and the representation of SANE, including the necessary knowledge and metadata. The basic prerequisites for the representation of semantic annotations are:

- An ontology, defining the entity classes and allowing unambiguous references to those;
- Entity identifiers, which allow these to be distinguished and linked to their semantic descriptions;
- A knowledge base (KB) with entity descriptions.

Entity descriptions actually make up the non-ontological part of formal knowledge in the semantic repository. The entity descriptions represent a KB, a body of instance knowledge or data. Such KB can either be available as pre-populated background knowledge and/or be extended through information extraction from the documents.

As with other sorts of annotations, a major question about the representation is: 'To embed or not to embed?' There are a number of arguments, giving evidence that semantic annotations are best decoupled from the content they refer to. One key reason for this is the ambition to allow for dynamic, user-specific, semantic annotations – conversely, embedded annotations become a part of the content and may not change according to the interest of the user or to the context of usage. Further, complex embedded annotations would have a negative impact on the volume of the content and could complicate its maintenance – e.g. imagine that a page with three layers of overlapping semantic annotations needs to be updated without compromising their consistency.

Given that semantic annotations should preferably be kept separate from the content to which they refer, the next question is whether or not (or to what extent) the annotations should be integrated with the ontology and the KB. It is the case that such an integration seems profitable – it would be easier to keep the annotation in sync with the class and entity descriptions. However, there are at least three important considerations to be made in this regard:

[18] Information extraction (IE) is a relatively young discipline within Natural Language Processing (NLP), which conducts partial analysis of text in order to extract specific information, (Cunningham 1999). IE and named entities are discussed in greater detail in Section 3 of Chapter 10.

[19] WordNet is the most popular large scale lexical database, providing partial descriptions of the word senses in the English language. It can be considered also as a lexical ontology or a KB. See the discussion around WordNet and its usage for semantic annotation and IR in Section 2 of Chapter 10.

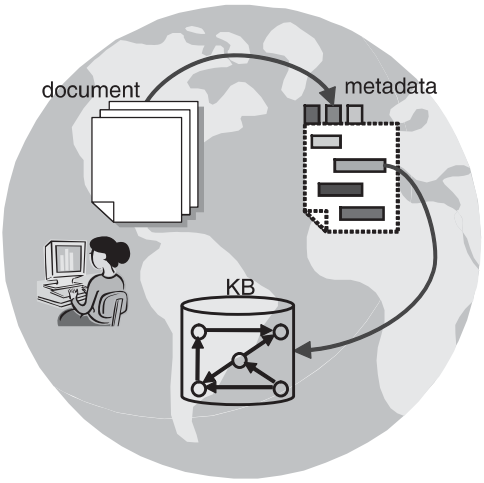

Figure 9.9 Distributed Heterogeneous Knowledge

- *Number and the complexity of the annotations*: these differ from those of the entity descriptions – the annotations are simpler, but more numerous than the entity descriptions. Even considering middle-sized corpora of documents, the number of annotations typically reaches tens of millions. Suppose that 10 M annotations are stored in an RDF(S) store together with 1 M entity descriptions. Suppose also that on average annotations and entity descriptions are represented with 10 statements each. The difference, regarding the inference approaches and the hardware that is capable of efficient reasoning and access to a 10 M-statement semantic repository and to a 110 M-statement repository, is considerable.
- *Separation of concerns*: if the world knowledge (ontology and instance data) and the document-related metadata are kept independent, this would mean that for one and the same document, different extraction, processing, or authoring methods will be able to deliver alternative metadata, referring to one and the same knowledge store.
- *Distributivity*: importantly, it should be possible that the ownership and the responsibility for the metadata and the knowledge are distributed. In this way, different parties can develop and separately maintain the content, the metadata, and the knowledge.

On the basis of these arguments, we propose a general model which allows for decoupled representation and management of the documents, the metadata (annotations), and the formal knowledge (ontologies and instance data), as illustrated in Figure 9.9

9.6 Semantic Indexing and Retrieval

As already mentioned, one of the key tasks which can be performed on top of semantic annotations is indexing and retrieval of documents with respect to the corresponding semantic features. This is a modification of the classical IR task – documents are retrieved on the basis of relevance to concepts instead of words. However the basic model is quite similar – a document is characterised by the bag of tokens[20] which constitute its content, disregarding its structure. While the basic IR approach considers

[20] Sometimes 'token' is used with a specialized meaning in the NLP field – in essence, tokens are the elements of the text, as they are separated by white spaces and punctuation (the delimiters are also considered tokens). (Kiryakov and Simov (1999) introduce the term 'atomic text entities' (ATE) as a general notion of token in IR context, to avoid ambiguity with the NLP usage of 'token'.

the word lemmata (base forms) or stems as tokens, there have been considerable efforts for the last decade related to indexing with respect to two sorts of higher-level semantic features, namely:

- Word-senses, lexical concepts, references to controlled vocabularies;
- Named entities (including numbers, dates, etc.).

Both types of indexing can serve as a basis for cross-language IR (see Chapter 11). Lexical concepts in one language can be related to such in another language or to some sort of interlingua – this was one of the main objectives for the development of the EuroWordNet[21] lexical ontology. On the other hand, once properly recognised, the named entities references are language independent (both the mentions of 'London' and 'Llundain' will be tagged with one and the same entity identifier).

9.6.1 Indexing with respect to lexical concepts

Many of the words in natural languages are polysemous, i.e. they can have more than one meaning. For instance, the word 'bank' can denote both a financial institution and a river bank. In WordNet (and other lexical ontologies) this linguistic phenomena is handled through association of the word with different lexical concepts[22] representing their different meanings. One of the main problems in the course of semantic annotation with respect to lexical ontologies is word-sense disambiguation (WSD) – the selection of the correct lexical concept, which represents the meaning of the word in the specific context. Once WSD is performed and documents are annotated with respect to lexical concepts, indexing and retrieval with respect to them solves the problem with query term polysemy and index term synonymy mentioned in Section 9.1.1. WSD and the usage of lexical resources for IR are discussed in Chapter 10. Here we will comment only on the principle advantages that such indexing can provide, given a lexical ontology with super-concept/sub-concept relationships, as well as possible indexing and retrieval techniques, as discussed in Kiryakov and Simov (1999). Suppose the following IR setup with hierarchically structured feature space:

- The documents are indexed with respect to (occurrence in the documents of references to) lexical concepts.
- The lexical concepts are properly related to the corresponding more general concepts (hypernyms) and less general concepts (hyponyms).
- The query is specified through lexical concepts (either because the user directly selected them, or because a 'bag of keywords' query has been semantically annotated).

Let us define *hyponym-matching* as a retrieval operator which matches more general concepts in the query with more specific ones in the document. Using such an operator a query containing the concept for 'bird' will match documents referring to 'duck' or 'eagle'; this follows the intuition that if a specific species of bird is mentioned, then this is also a reference to bird. On the contrary, a document which refers to 'bird' will not be hyponym-matched to a query including 'hen'; intuitively, there is a no guarantee that a document about birds has something to say about 'hens'.

First, let us note that hyponym-matching offers clear benefits for the users as compared with the mainstream IR techniques (e.g. vector-space model using word lemmata as features). In a typical search engine, documents mentioning 'duck', but not mentioning explicitly 'bird', will not be returned as a result for a query for 'bird' – the lack of hyponym-matching leads to poor recall. To fix this 'manually' the user should include in the query all possible species of birds and their synonyms, which would be an unreliable and inefficient exercise. Further, the words from the expanded query will match words in the text, which may be used in a different meaning there – for example, the query expansion will

[21] http://www.illc.uva.nl/EuroWordNet/

[22] The lexical concepts in WordNet are called synsets (from synonym set, as already mentioned).

contain 'duck' which in a specific text could have been used for cloth[23]. The effect is a reduction in precision.

An interesting question is how hyponym-matching can be integrated into an existing system, e.g. a vector-space-based probabilistic model. Let us see first how indexing and retrieval can be performed with respect to lexical concepts (disregarding hyponymy). Suppose that before being indexed the documents are pre-processed as follows: (i) they are semantically annotated with lexical concepts; and (ii) each occurrence of a word (or a multi-word token) is replaced by the identifier of the corresponding concept. The queries can be pre-processed in the same manner. In a simplified world, this is an easy way to make an existing IR engine implement semantic search – the vector-based similarity between queries and documents should be a good model for relevance, as when documents are indexed with respect to word lemmata.

One straightforward solution for extending this model with hyponym-matching is query expansion. Each of the concepts in the query can be replaced with the set of itself plus all of its hyponyms (sub-concepts). This approach is simple and can prove sufficient in many contexts, but one should be aware of its disadvantages:

- A concept with a bigger set of hyponyms will gain bigger weight in the relevance calculation as opposed to such with no or just a few hyponyms. This problem can partly be solved if the IR engine supports disjunction (i.e. OR operator) – however, this is not a natural feature for the engines based on probabilistic models.
- The set of all sub-concepts of all the concepts in the query, could grow to thousands of elements, which can cause problems with the performance of the IR engine.

An alternative approach (let us name it *hypernyms-indexing*) is to modify the indexing strategy, so, that the documents are indexed with respect to the hypernyms of the concepts they refer to. In such case, a document mentioning the concept 'eagle' will also appear in the reverse index for its super-concept (e.g. 'bird') and the super-concept of the super-concept (e.g. 'animal') and so forth following the subsumption chain to the top concept. In cases when there is no control of the engine's indexing strategy, this effect can be achieved if each word gets annotated not only with the specific lexical concept, corresponding to its meanings, but also with the super concepts. Then in the document pre-processing phase, the identifiers of the super-concepts will also appear in the document index.

Query expansion is no longer necessary, when hypernym-indexing is involved, because the relevance of the document to the more general concepts has been reflected in the indices. A query for 'bird' will retrieve a document which only mentions 'eagle', without any need for query modification. The problems of this approach can be summarised as follows:

- The overall size of the indices will grow, due to the fact that each token in the document appears in multiple indices. The growth can be estimated as a factor close to the average depth of the of the hypernymy hierarchy;
- The indices for the most general concepts will get huge and cause efficiency problems.

These problems can be addressed to some extent if limited query- or index-expansion is performed. For instance, one can put a constraint on the number of levels of the hypernyms to be considered or to the total number of hyponyms to be used for expansion. In all cases, it should be clear that the adaptation of a probabilistic IR model (tuned for a flat feature set) to deliver good performance for a hierarchically structured feature set is far from trivial. Experiments show that the adaptation of a "general-purpose" lexical ontology for this task is problematic, (Voorhees 1998). An interesting implementation is reported in (Mahesh *et al*. 1999): a large-scale lexical ontology, built for the specific IR task, is used for the IR engine built into one of the major RDBMS systems. The evaluation of the system on some of the tasks of the TREC competition, prove clear performance benefits for this approach.

[23] According to WordNet 2.1 (http://wordnet.princeton.edu/perl/webwn) one of the meanings of 'duck' is: a heavy cotton fabric of plain weave; used for clothing and tents.

9.6.2 *Indexing with respect to named entities*

A recent large-scale human interaction study on a personal content IR system of Microsoft, (Dumais *et al*. 2003), demonstrates that, at least in some cases, named entities are central to user needs:

> The most common query types in our logs were People/Places/Things, Computers/Internet, and Health/Science. In the People/Places/Things category, names were especially prevalent. Their importance is highlighted by the fact that 25% of the queries involved people's names, which suggests that people are a powerful memory cue for personal content. In contrast, general informational queries are less prevalent.

As the volume of web content grows rapidly, the demand for more advanced retrieval methods increases accordingly. Based on semantic annotation of named entities (SANE), efficient indexing and retrieval techniques can be developed, involving an explicit handling of the named entity references.

In a nutshell, SANE could be used to index both 'NY' and 'N.Y.' as occurrences of the specific entity 'New York', as though a unique identifier for that entity occurred in the text in place of the different syntactic variations of the strings used to denote it. Since many present systems do not involve entity recognition, they will index on 'NY' (for the former), and 'N' and 'Y' (for the latter), which demonstrates well some of the problems with the keyword-based search engines. It should be noted that there have been previous attempts to conflate synonymous terms to the same unique index token (e.g. Robertson *et al*. 1991), though these have tended to lack the natural language processing technology deployed more recently. A survey of recent approaches to semantic annotation and human language technology can be found in Bontcheva *et al*. (2006).

Given metadata-based indexing of content, advanced semantic querying becomes feasible. A query against a repository of semantically annotated documents can be specified in terms of restrictions on the entity's type, name, attribute values, and relations to other entities. For instance, a query can request all documents that refer to Person-s that hold some Position-s within an Organisation, and which also restricts the names of the entities or some of their attributes (e.g. a person's gender). Further, semantic annotations can be used to match specific references in the text to more general queries. For instance, a query such as 'Redwood Shores company' could match documents mentioning specific companies such as ORACLE and Symbian, which are located in this town.

Hybrid query modes, such as the one mentioned above, can provide unmatched analytical levels through a combination of:

- Database-like structured queries, extended with the reasoning capabilities of the semantic repositories;
- IR-like probabilistic models.

Although the above sketched enhancements look promising, further research and experimentation are required to determine to what extent and in which way(s) they can improve existing IR systems. It is hard, in a general context, to predict how semantic indexing will combine with the symbolic and the statistical methods currently in use. Large-scale experimental data and evaluation efforts (similar to TREC) are required for this purpose.

9.6.3 *Retrieval as spreading activation over semantic network*

Rocha *et al*. (2004) describe a search architecture that applies a combination of spread activation and conventional information retrieval to a domain-specific semantic model in order to find concepts relevant to a keyword-based query. The essence of spread activation, as applied in conventional textual searching, is that a document may be returned by a query, even if it contains none of the query keywords. This happens if the document is linked to by many other documents which do contain the keywords.

In the case described here, the user expresses the query as a set of keywords. This query is forwarded to a conventional search engine which assigns a score to each document in its index in the usual way. In addition to a conventional index, the system contains a domain-specific KB, which includes instance nodes pointing to web resources. As usual in RDF, each instance is described in terms of links labelled with properties in compliance with the ontology. The basic assumption is that weightings that express the strength of each instance relation can be derived[24]. Thus the resulting network has, for each relationship, a semantic label and an associated numerical weight. The intuition behind this approach is that better search results can be obtained by exploiting not only the relationships within the ontology, but also the strength of those relationships.

Searching proceeds in two phases: as mentioned, a traditional approach is first used to derive a set of documents from a keyword-based query. As discussed, these documents are annotated with instances nodes – this set of nodes is supplied as an initial set to the spread activation algorithm, using the numeric ranking from the traditional retrieval algorithm as the initial activation value for each node. The set of nodes obtained at the end of the propagation are then presented as the search results. Two case studies are reported, showing how in some cases the combination of the traditional and spread activation techniques performs better than either on its own.

9.7 Semantic Search Tools

This section presents several tools which adopt different approaches for semantic search. These represent concrete implementations which demonstrate applications of semantic search and semantic annotation:

(i) QuizRDF was a relatively early semantic search tool, supporting search over document-level annotations expressed in RDF(S);
(ii) TAP augments conventional search results with relevant information aggregated from distributed semantic information sources;
(iii) KIM is a platform providing infrastructure and services for automatic semantic annotation, indexing, and retrieval of unstructured and semi-structured content;
(iv) Squirrel builds on KIM and a number of other component technologies to provide an advanced semantic search engine.

9.7.1 Searching through document-level RDF annotations – QuizRDF

Even the simplest semantic Web standard, RDF(S), allows for content annotation in ways much more flexible than those of the syntactic annotation formalisms, such as XML:

- RDF(S) is descriptive not prescriptive – XML dictates the format of individual documents; whereas RDF(S) allows the description of any content and the RDF(S) annotations need not be embedded within the content itself.
- More than one RDF(S) ontology, schema, or dataset can be combined to describe the same content; there could be alternative annotations for one and the same document, provided from different sources for different purposes.
- RDF(S) has a well-defined semantics regarding, for example, the sub-class relation; it allows the definition of a set of relations between resources as described in Sections 9.2.2 and 9.4.

The QuizRDF search engine (Davies *et al*. 2003) combines free-text search with a capability to exploit RDF annotation querying. QuizRDF combines search and browsing capabilities into a single

[24] A number of different approaches to this derivation are taken and the authors state that no single weight derivation formula is optimal for all application areas.

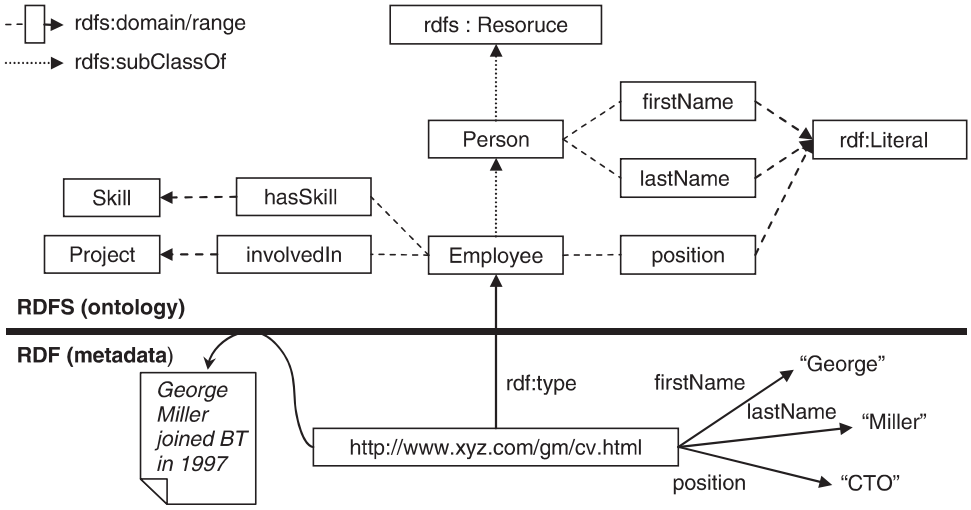

Figure 9.10 Semantic Annotation Datamodel in QuizRDF

tool and allows document-level RDF annotations to be exploited (searched over) where they exist, but will still function as a conventional search engine in the absence of those annotations

A simplified example of semantic annotation in QuizRDF is given in figure 9.10. The data model employed for semantic annotation and the indexing scheme of QuizRDF can be summarized as follows:

- The ontology is represented as an RDF schema. According to the classification in Section 9.1.2, QuizRDF assumes a schema-ontology, e.g. there are classes (such as Person) with their appropriate attributes (properties with literal values, e.g. lastName) and relations to other classes (e.g. hasSkill), which are being instantiated.
- Each document is linked to one or more subjects via the rdf:type property. The RDF resource (identifying the document) is considered to be an instance of the class representing the subject. This way, the CV of a person and the person herself are modelled with a single RDF resource[25].
- A document can be described with properties appropriate for the class of its subject. For example. a document associated with subject Employee can be given attributes firstName, lastName (inherited from the Person super-class) and position.
- Full-text indexing is applied not only to the content of the documents, but also the literal values of any relations. This way a keyword search for 'CTO' will return Miller's CV document, although this string may not appear in the document itself[26].

This data model would have been clearer if the documents and the objects they are about were defined as separate RDF resources, e.g. <cv,isAbout,emp1>, <emp1,type,Employee>, <emp1,firstName,'George'>. On the other hand, this model has all the advantages of being more simple and efficient.

The user enters a query into QuizRDF as they would a conventional search engine. A list of documents is returned, ranked according to the resource's relevance to the user's query using a

[25] Note that the modelling approach described in identifying a web page (CV) as an instance of class Employee could be seen as epistemologically naïve: a web page is a document, not a person. In a later version of QuizRDF, a more sophisticated approach is taken and is discussed in Davies *et al.* (2003).

[26] The information about the position of George Miller could have been acquired as background knowledge added to the KB from some database or extracted from another document. The usage of background knowledge is discussed in the next section.

traditional vector space approach (Salton *et al*. 1975). The subjects of the documents returned in response to a query are ascertained and displayed along with the traditional ranked list. By selecting one of the displayed subjects, the user can filter the retrieval list to include only those documents associated with it. QuizRDF also displays the properties and classes related to the selected class. Each class displayed has an associated hyperlink that allows the user to browse the RDF(S) ontology: clicking on the class name refreshes the display to show that class properties and related classes[27] and again filters the results list to show only URIs of the related class. Where properties have literal types (e.g. string) as their range, QuizRDF enables users to query against these properties. An example of search of the documents about the employee George Miller is given in Figure 9.11; Note that, based on the ontology from Figure 9.10, one might imagine there would be a fourth field labelled 'position' – which relations are displayed for instances of a given class is a configurable parameter in QuizRDF and in this case, the relation 'position' has not been selected for display. This can be important in an ontology where classes have many relations, some perhaps inherited from other classes). Another example, would be a search for documents associated with class Painting, which has a property hasTechnique with range of type 'string'. If a set of documents has been returned of class Painting, a user could enter 'oil on canvas' as a value for hasTechnique and the document list would be filtered to show only those documents which have the value 'oil on canvas' for this property.

Thus QuizRDF has two retrieval channels: a keyword query against the text, and a much more focussed query against specific RDF properties, as well as supporting ontology browsing. Searching the full text of the documents ensures the desired high recall in the initial stages of the information seeking process. In the later stages of the search, more emphasis can be put on searching the RDF annotations (property values) to improve the precision of the search.

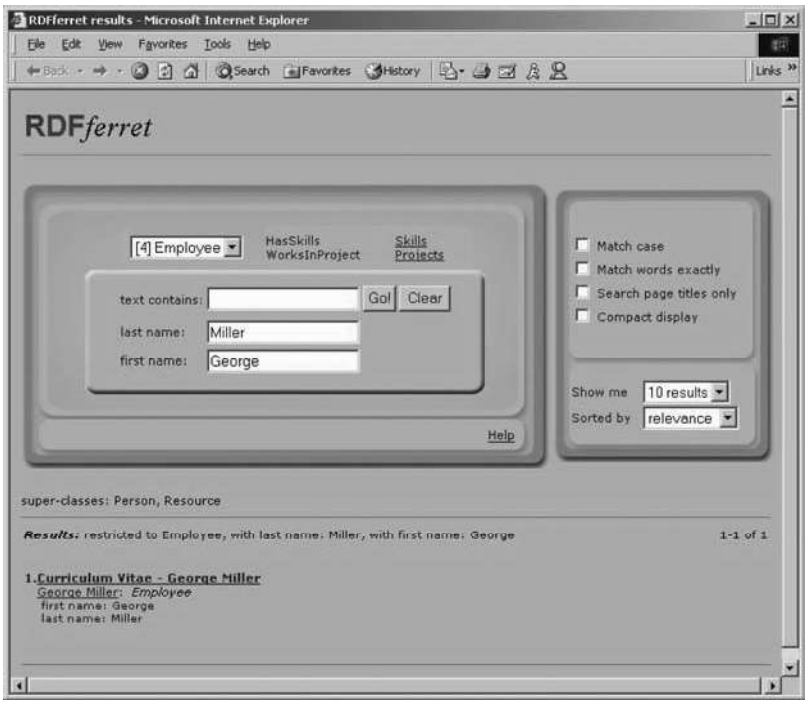

Figure 9.11 QuizRDF Search Interface (screen shot from an early version named RDFferret) (Reproduced by Permission of © 2009 British Telecommuncations Plc)

[27] A related class in this context is a class which is the domain or range of a property of the original class.

Experiments with QuizRDF on an RDF-annotated web site showed improvements in performance as compared to a conventional keyword-based search facility (Iosif *et al*. 2003).

9.7.2 *Exploiting massive background knowledge – TAP*

As discussed above, conventional search engines have no model of how the concepts denoted by query terms may be linked semantically. When searching for a paper published by a particular author, for example, it may be helpful to retrieve additional information that relates to that author, such as other publications, curriculum vitae, contact details, etc. A number of search engines are now emerging that use techniques to apply ontology-based domain-specific knowledge to the indexing, similarity evaluation, results augmentation and query enrichment processes.

TAP, (Guha and McCool 2003), is Semantic Web architecture, which allows RDF(S)-compliant consolidation and querying of structured information. Guha *et al*. (2003) describe a couple of Semantic Web-based search engines: ABS – activity-based search and W3C Semantic Search. In both cases TAP is employed to improve traditional search results (obtained from Google, http://www.google.com) when seeking information in relation to people, places, events, news items, etc. TAP is used for two tasks:

- Result augmentation: the list of documents returned by the IR system is complemented by text and links generated from the available background knowledge.
- Query term disambiguation: the user is given the opportunity to choose the concrete entity she is searching for, then the system attempts to filter the results of the IR system to those referring only to this entity. An approach using several statistics for this purpose is sketched in (Guha *et al*. 2003) without details on the implementation.

The semantic search application, which runs as a client of TAP, sends a user-supplied query to a conventional search engine. Results returned from the conventional search engine are augmented with relevant information aggregated from distributed data sources that form a knowledge base (the information is extracted from relevant content on targeted websites and stored as machine-readable RDF annotations). The information contained in the knowledge base is independent of and additional to the results returned from the conventional search engine. A search for a musician's name, for example, would augment the list of matching results from the conventional search engine with information such as current tour dates, discography, biography, etc. Figure 9.12 shows a typical search result from ABS. Special attention is paid to the selection of a dataset to show and its presentation.

Relatively simple heuristics are used to find the concepts which are relevant to the query. No considerable processing of the query terms is performed – according to Guha *et al*. (2003), concepts are considered relevant if they have a label (name) that contains one of the query terms.

The couple of semantic search engines mentioned above do not perform any pre-processing or indexing of the documents – they are based on an existing search engine. Dill *et al*. (2004) presents a system called SemTag, which performs automatic semantic annotation of texts with respect to large scale knowledge bases available though TAP, solving a task similar to the one presented in the next section.

9.7.3 *Character-level annotations and massive world knowledge – KIM*

The KIM platform (Popov *et al*. 2003), provides infrastructure and services for automatic semantic annotation, indexing, and retrieval of unstructured and semi-structured content.

As a baseline, KIM performs character-level semantic annotation of named entities, as described in Section 9.5. The automation of this task is possible through information extraction technology, which the General Architecture for Text Engineering (GATE) (http://www.gate.ac.uk). KIM analyzes texts and recognizes references to entities (such as persons, organizations, locations, dates). Then it tries to match the reference with a known entity that has a unique URI and description. In cases when

Figure 9.12 Semantic Search with TAP

there is no match, a new URI and description are generated automatically – this is the situation when ontology population takes place. Finally, the reference in the document is annotated with the URI of the entity, as presented in Figure 9.8.

KIM is equipped with an Internet Explorer plug-in, which uses these annotations for highlighting and hyperlinking, as presented in Figure 9.13. The mentions are coloured in accordance with the class (type) of the entity; hyperlinks provide access to popup forms presenting their descriptions in the KB. As the later include also references to other related entities, the user can further traverse the KB. This way the plug-in allows smooth transition from the text to the KB and exploration of the available structured knowledge.

In order to enable the easy bootstrapping of applications, KIM is based on the PROTON ontology (Terziev *et al*. 2004), which consists of about 250 classes and 100 properties. Furthermore, a knowledge base (KIM's World KB, WKB), pre-populated with about 200 000 entity descriptions, is bundled with KIM. Its role is to provide as a background knowledge (resembling a human's common culture) a quasi-exhaustive coverage of the entities of general importance – those, which are considered well-known and thus not explicitly introduced in the documents, which makes it hard to get their descriptions automatically extracted. KIM uses the OWLIM high-performance semantic repository (http://www.ontotext.com/owlim) to manage the WKB together with the extracted instance data.

A unique entity ID is inserted in the text at the places where the entity is referred. The application of entity co-reference resolution means that the system would regard the strings 'Gordon Brown,' 'Mr Brown,' 'the Prime Minister' as referring to the same entity in the KB. Then the texts are passed for indexing to a standard full-text search engine; in its basic configuration KIM uses for this purpose the Lucene engine (http://lucene.apache.org/java/docs/).

This allows KIM to offer the semantic queries which combine structured queries, reasoning, and full-text search. The most generic search interface (named Entity Pattern Search), allows the specification of queries about any type of entity, relations between such entities and required attribute values (e.g. 'find' all documents referring to a Person that hasPosition 'CEO' within a Company, locatedIn

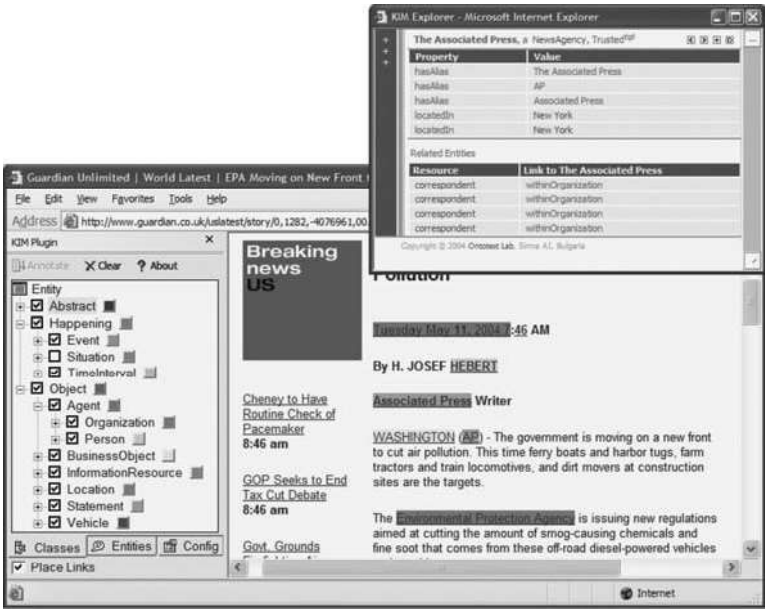

Figure 9.13 Semantic Browsing and Navigation in KIM (Reproduced by Permission of Ontotext AD)

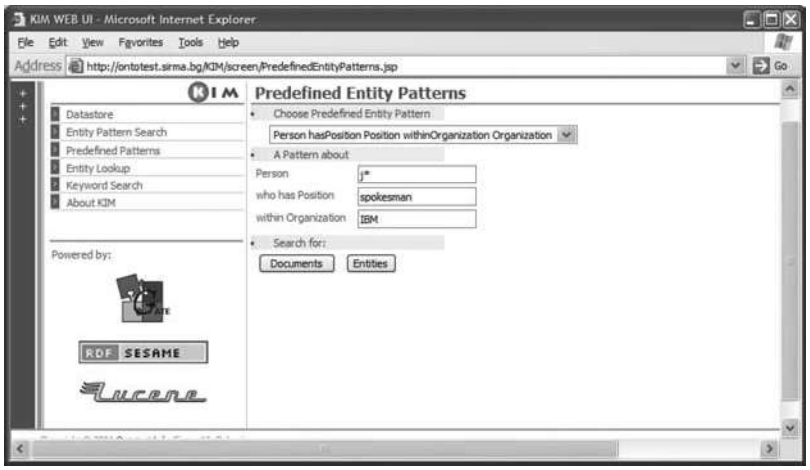

Figure 9.14 Semantic Querying in KIM (Reproduced by Permission of Ontotext AD)

a Country with name 'UK'). To answer the query, KIM applies the semantic restrictions over the entities in the KB. The resulting set of entities is matched against the semantic index and the referring documents are retrieved with relevance ranking according to these entities.

KIM provides also a simplified search interface for several predefined patterns. In Figure 9.14 a semantic query is specified, concerning a person whose name begins with J, and who is a spokesman for IBM.

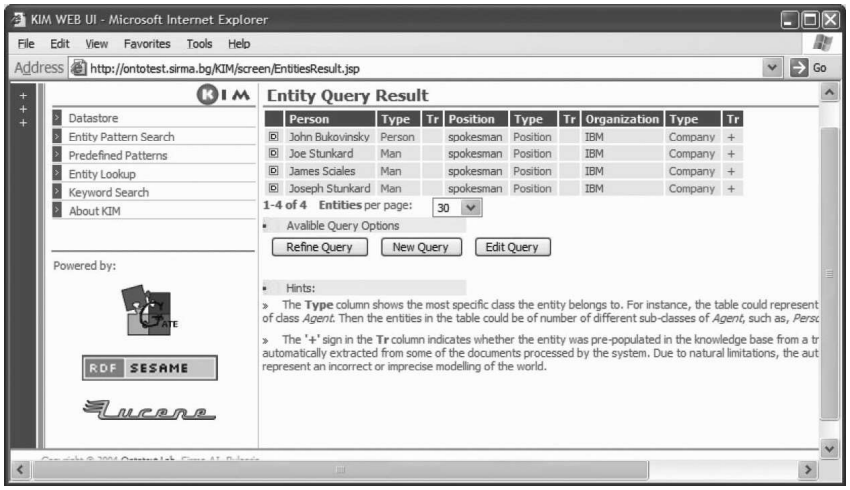

Figure 9.15 Semantic Query Results (Reproduced by Permission of Ontotext AD)

Figure 9.15 shows that four entities have been found in the documents indexed. It is then possible to browse a list of documents containing the specified entities and KIM renders the documents, with entities from the query highlighted (in this example IBM and the identified spokesperson).

In other work, Bernstein *et al*. (2005) describe a controlled language approach whereby a subset of English is entered by the user as a query and is then mapped into a semantic query via a discourse representation structure. Vallet *et al*. (2005) propose an ontology-based information retrieval model using a semantic indexing scheme based on annotation weighting techniques.

9.7.4 Squirrel

Squirrel (Duke *et al*. 2007) provides combined keyword-based and semantic searching. The intention is to provide a balance between the speed and ease of use of simple free text search and the power of semantic search. In addition, the ontological approach provides the user with a rich browsing experience. Squirrel builds on and integrates a number of semantic technology components:

(i) PROTON, the lightweight ontology and world knowledge base discussed above is used, and user profiles reflecting end-user personal interests are modelled in PROTON.
(ii) Lucene[28] is used for full-text indexing.
(iii) The KAON2 (Motik and Studer 2005) ontology management and inference engine provides an API for the management of OWL-DL and an inference engine for answering conjunctive queries expressed using the SPARQL[29] syntax. KAON2 also supports the Description Logic-safe subset of the Semantic Web Rule Language[30] (SWRL). This allows 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.
(iv) OntoSum (Bontcheva 2005), a Natural Language Generation (NLG) tool, takes structured data in a knowledge base as input and produces natural language text, tailored to the presentational

[28] http://lucene.apache.org/
[29] http://www.w3.org/TR/rdf-sparql-query/
[30] http://www.w3.org/Submission/SWRL/

context and the target reader. In the context of the Semantic Web and knowledge management, NLG is required to provide automated documentation of ontologies and knowledge bases and to present structured information in a user-friendly way. 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 natural language summary of information regarding those entities. In addition, document results can be enhanced with brief descriptions of key entities that occur in the text.

(v) KIM (see above) is used for massive semantic annotation.

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

The first task Squirrel carries out after the user submits a search is to call the Lucene index and then use KAON2 to look up further details about the results, be they 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, e.g. a search for 'Airline Industry' would match the 'Airline' class in PROTON. Selecting this would then allow users 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 content items can by separated by their type, e.g. 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.

9.7.4.2 Meta-result

The meta-result page is intended to allow users 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 Figure 9.16 under the subheading '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 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 Figure 9.16) and display them on their own line in the meta-result.

9.7.4.3 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.

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

Figure 9.16 Meta-result

the meta-result section. Instead of showing the documents of the topic, the metadata 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 metadata is shown and what is not. The use of these templates is discussed further in Section 9.7.4.6 where a 'Company' entity view is described.

9.7.4.4 Attribute-based refinement

Any 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 downside to 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 size of the result set once constraints have been set. The user can then modify them before asking Squirrel to build the result page.

9.7.4.5 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 metadata 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 markup 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. A screenshot of the document view is shown in Figure 9.17.

The document view also shows the whole document abstract, marked up with entity annotations. 'Mousing-over' these entities provides the user with further information about the entity extracted from the ontology. Clicking on the entity itself takes the user to the entity view.

9.7.4.6 Entity view

The entity view for 'Sun Microsystems' is shown in Figure 9.18. It includes a summary generated by OntoSum. 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

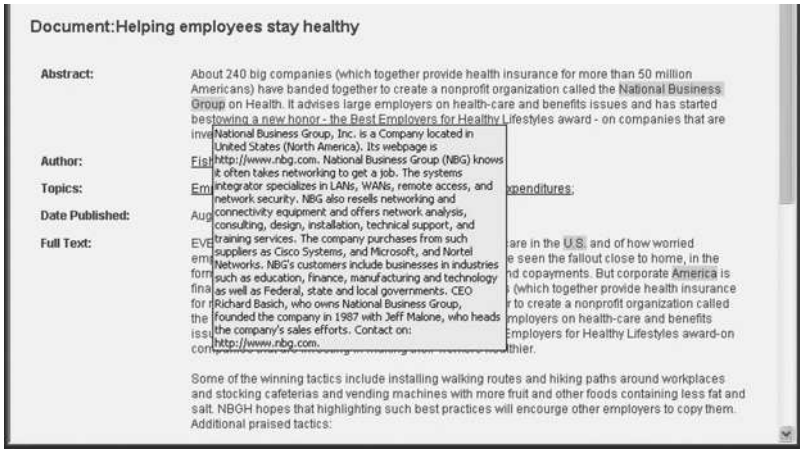

Figure 9.17 Document view

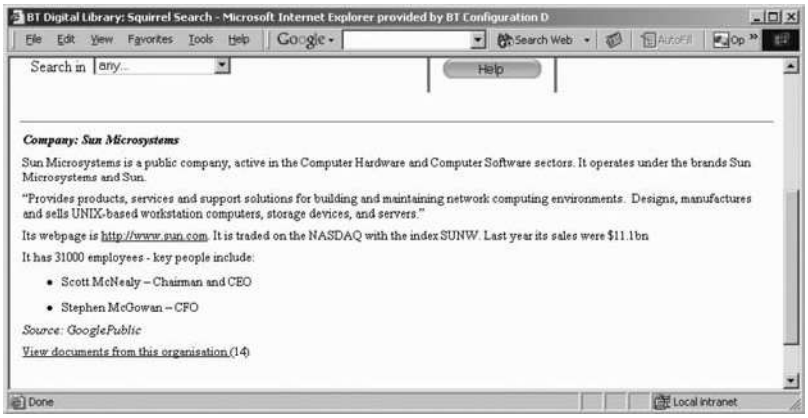

Figure 9.18 Company entity view

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.

9.7.4.7 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 9.19 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 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 checkboxes.

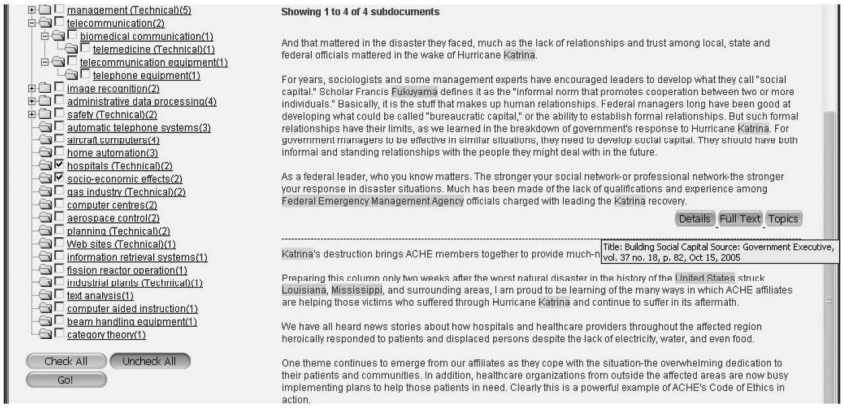

Figure 9.19 Consolidated results

9.7.4.8 Evaluation

Squirrel has been subjected to a three-stage user-centred evaluation process with users of a large digital library. Results are promising regarding the perceived information quality (PIQ) of search results obtained by the subjects. From 20 subjects, using a 7 point scale the average (PIQ) using the existing library system was 3.99 compared with an average of 4.47 using Squirrel – a 12% increase. The evaluation also showed that users rate the application positively and believe that it has attractive properties. Further details can be found in Thurlow and Warren (2008).

9.7.5 Other approaches

In this section, we briefly mention some other systems which have been described in the published literature: note that this list – along with the systems described above – is intended to be indicative, rather than exhaustive.

9.7.5.1 Searching for semantic web resources

We have seen in the earlier sections a variety of approaches to searching XML, RDF and OWL annotated information resources. The Swoogle search engine (Ding *et al*. 2004) is tackling a related but different problem: it is primarily concerned with finding ontologies and related instance data.

Finding ontologies is seen as important to avoid the creation of new ontologies where serviceable ones already exist, thus, it is hoped, leading to the emergence of widely used canonical (or reference) ontologies. Swoogle supports querying for ontologies containing specified terms. This can be refined to find ontologies where such terms occur as classes or properties, or to find ontologies that are in some sense about the specified term (as determined by Swoogle's ontology retrieval engine). The ontologies thus found are ranked according to Swoogle's OntologyRank algorithm which attempts to measure the degree to which a given ontology is used.

In order to offer such search facilities, Swoogle builds an index of semantic web documents (defined as web-accessible documents written in a Semantic Web language). A specialised crawler has been built using a range of heuristics to identify and index semantic web documents.

The creators of Swoogle are building an ontology dictionary based on the ontologies discovered by Swoogle[31].

[31] http://swoogle.umbc.edu/

9.7.5.2 Semantic browsing and navigation

Web browsing complements searching as an important aspect of information-seeking behaviour. Browsing can be enhanced by the exploitation of semantic annotations and below we describe systems which offer a semantic approach to information browsing (in some cases combined with searching).

Magpie (Domingue *et al*. 2004) is an internet browser plug-in which assists users in the analysis of webpages. Magpie adds an ontology based semantic layer onto web pages on-the-fly as they are browsed. The system automatically highlights key items of interest (in a way similar to KIM, see Figure 9.13), and for each highlighted term it provides a set of 'services' (e.g. contact details, current projects, related people) when you right-click on the item. This relies, of course, on the availability of a domain ontology appropriate to the page being browsed – similarly to TAP, Magpie annotates on the basis of match of a label from the KB.

CS AKTiveSpace (Glaser *et al*. 2004) is a Semantic Web application which provides a way to browse information about the UK Computer Science Research domain, by exploiting information from a variety of sources, including funding agencies and individual researchers. The application exploits a wide range of semantically heterogeneous and distributed content. AKTiveSpace retrieves information related to almost 2000 active computer science researchers and over 24 000 research projects, with information being contained within thousands of published papers, located in various university websites. This content is gathered on a continuous basis using a variety of methods, including harvesting publicly available data from institutional websites, bulk translation from existing databases, as well as other data sources. The content is mediated through an ontology, and stored as RDF triples; the indexed information comprises around 10 000 000 RDF triples in total.

CS AKTive Space supports the exploration of patterns and implications inherent in the content, using a variety of visualisations and multi-dimensional representations to give unified access to information gathered from a range of heterogeneous sources.

Alonso (2006) presents an impressive proof-of-concept prototype of a KM solution, using various features of ORACLE 10gR2 to implement semantic metadata-based search and browse. The system combines RDF support, full-text indexing, and clustering functionality. The system's user interface implements several navigation modes, based on visualisation components available as libraries from third parties.

Siderean's Seamark Navigator[32] allows for faceted search based on document-level metadata represented in RDF: subject, author, publisher, date. Internally, the system combines an RDF repository (which may or may not incorporate reasoning) and a full-text search engine. The search user interface allows for interactive focussing: for each facet the user is presented with the most popular values. For each value (e.g. a specific author) the system presents the number of documents matching this value – this provides information to the user about the 'selectivity' of the specific values. The user can choose a value of a facet and it is added to the filter, following which the values and the selectivity figures presented are altered to consider only the documents matching the current filter.

9.8 Summary

In this chapter, we began by discussing some of the limitations of current search technology and proceeded to discuss ontologies and Semantic Web technology and how they could be used to annotate document collections semantically, particularly with regard to named entities occurring within documents. We then explained how semantic annotations could be used in systems to overcome some of the problems we earlier identified with respect to current search technology. Finally, a number of systems using semantic search techniques were described. In general, these systems allow a greater precision in search and can also link entities occurring in document collections to ontologies, effectively providing a 'background knowledge base' to augment search results. Furthermore, the formal nature of the ontological KBs means that it is possible to reason over them (taking a simple example, if searching for an organisation in France, an organisation in Paris might be returned even if it is

[32] http://www.siderean.com/products_suite.aspx)

nowhere in the document collection linked explicitly to France). Of course, the cost of generation and maintenance of such KBs and of semantic annotation of documents need to be weighed against the potential benefits of such an approach for each potential domain of application. It is worth noting that the automatic or semi-automatic generation of ontologies and semantic annotations is a focus for much current research (see, for example, Davies *et al*. 2009), but space has not permitted a detailed discussion of the prospects for such technology in this chapter.

Basic definitions regarding ontologies can be found in Gruber (1992; 1993) Guarino and Giaretta (1995) and Guarino (1998).

Those wishing to find out more about semantic technology are referred to (Davies et al. 2006).

Acknowledgements. Paul Warren is thanked for his comments on a draft of this chapter.

Exercises

9.1 Search engine users have been found to enter only a small number of query terms. Outline two difficulties arising from this fact associated with current search engine technology. Explain briefly how the semantic search approach aims to overcome these.

9.2 Given the ontology shown in Figure 9.8, explain how a query containing the terms *London* and XYZ could match a document containing the terms *UK* and *Company*.