University of Liverpool - Technical Report January 2008

QuestSemantics
Intelligent Search and Retrieval of Business Knowledge

I. Blacoe, L. Ianonne, I. Palmisano and V. Tamma

Department of Computer Science, University of Liverpool
L69 3BX Liverpool UK
{I.W.Blacoe, L.Iannone, ignazio, V.A.M.Tamma}@csc.liv.ac.uk

Abstract. Keyword-based search engines, though hugely popular, are limited when trying to answer very specific queries. The processing of search results is performed by users, rather than the software. *Ontologies* provide a means to create formal, machine-processable descriptions of the knowledge in a certain domain [1], and by using elements of these descriptions to annotate suitable information sources, they can be analysed and manipulated in an *intelligent* manner. Our QuestSemantics platform provides automated, ontology-based metadata creation and resource annotation, with subsequent ontology-based querying of the annotated resources. This platform has been deployed in two commercial scenarios, providing useful feedback on both the feasibility and effectiveness of applying semantic web technologies to specific business problems.

1 Introduction

In today's Web the information is primarily intended to be read and processed by humans, and cannot be readily manipulated by computers. The intelligence applied in search tasks, as well as the assessment of the relevance of retrieved pages, is mainly human, with limited support from software [2]. Whilst this type of processing is still adequate for domestic users, it cannot scale to the volume of information available to business, where the vast amount of data available on the web is coupled to company documents and databases. Current keyword based search engines present limitations in that they cannot fully capture the the richness intrinsic in natural language; an example are the problems caused by synonymy and polysemy to the search task. Enhancing search engines with lexicons such as WordNet [3] can help to relieve the problem, but is not sufficient to identify and resolve more complicated types of ambiguity. Furthermore, keyword-based search engines make little provision for the formulation of very specific queries, particularly those that make use of relationships between entities.

A possible way to overcome these limitations is to make use of semantic web technology. The semantic web [4] is an evolution of the current web where information is represented in a machine-readable format, while maintaining the human-friendly representation in HTML. The idea is to unleash the power of the web by extending it to become a container of interconnected, machine-processable information and knowledge, rather than just a collection of documents. This is made possible by placing an infrastructure over the web that allows the annotation of resources in order to explicitly represent their content. The value of knowledge representation is not limited to web pages alone, as the technology can also be applied to corporate databases, managed documents, multi-media resources and other information sources of various types that are now available on the web, but need to be interpreted by humans (a typical example would be a query for specific pictures on Google).

Ontologies [1] are crucial in providing shared and machine processable *meaning* to web resources. An ontology models the entities and processes that are used to describe the content of a web resource, and, most importantly, the logical relations existing between them. Using this model, a representation can be created of the information contained in relevant web documents (*annotation*), and thus more precise queries can be formulated to retrieve this information. Annotation is normally achieved by using or creating metadata items (as instances of concepts from the ontology) to represent specific entities recognised in the resources, and then linking this metadata to the resource as its description. Many research efforts have thus been devoted to the provision of (semi-) automatic solutions for annotating web documents expressed in various formats, mainly text, but also structured formats, such as databases.

In this paper we present QuestSemantics, a platform supporting the semi-automatic discovery, annotation, filtering and retrieval of information resources on the internet and in intranets, on the basis of fine-grained business knowledge. QuestSemantics is designed in order to maximise the separation between the different types of knowledge represented - domain versus task-specific knowledge, and application versus generic knowledge. This separation is aimed at achieving reusability, and easy customisation of the various architectural components, thus allowing semantics-based search in a variety of task and domain scenarios. The platform is comprised of two main components: a general framework for the (semi-) automatic annotation of resources, based upon a detailed ontological model of the domain, and a search interface for the user-friendly formulation and execution of knowledge-based queries over the generated metadata.

We have focused on applying knowledge representation and manipulation within restricted and clearly delimited domains of knowledge, enabling detailed conceptualisations that provide the means to discover, annotate, filter and search resources on the basis of fine-grained business knowledge. Once a suitable ontology of the domain knowledge has been constructed, it can be used to identify and semantically annotate relevant resources. This ontology also underpins the provision of fine-grained access to the identified knowledge resources on the basis of the domain knowledge. Thus, through the application of knowledge representation and manipulation techniques, within specific contexts, we can apply a degree of 'intelligence' to the problem of access to information.

The paper illustrates two different commercial use-cases in which the QuestSemantics platform has been employed, providing concrete data on the advantages that the adoption of semantic web technologies can bring to classical information retrieval problems. The remainder of the this paper is organised as follows. The next section, *Platform design and implementation*, describes the design and implementation of the developed application. Section 3, *Test-case deployment and evaluation*, gives details regarding the deployment and evaluation of the platform in two commercial test-cases, one in the safety legislation compliance contracts domain, the other in the aerospace domain. We then describe related work in Section 4, and in Section 3, *Conclusions*, we draw some conclusions based on our experiences in this project.

2 Platform design and implementation

The framework we present is designed for applications that aim to leverage different information sources in order to provide searchable knowledge. Such a requirement is often accomplished by means of steps that only differ slightly between different applications and different domains. The aim of our framework is to enable applications to abstract from all the details that are common, so that application specific code is reduced and simplified. In the reminder of this section we discuss the main aspects underlying the design and implementation of QuestSemantics.

2.1 Knowledge independent components

QuestSemantics is a generic platform for the automatic annotation and retrieval of semi-structured information sources based on semantic queries (i.e. queries that make use of knowledge about the application domain). The platform components are designed to be customisable depending on the specific domain it is applied to. Therefore, a main concern in the platform design is that its customisation regards only domain related aspects, i.e. an application of our platform is the result of providing *application specific knowledge* to the general framework. In our design we make a distinction between *domain knowledge* and *task knowledge*. Domain knowledge is the description of all relevant entities in a specific domain of knowledge, representing a state of affairs and constraining the possible states it can evolve into. Task knowledge, in general, references the domain knowledge to describe the relevant entities with respect to the required tasks [5], and thus describes the ways to perform useful changes to the domain states. Though separate, they are dependant on each other - in fact domain knowledge representation cannot be independent from the task it will be used for; this problem has been described in [6] as the *interaction problem*.

The only decisions taken at platform level are those related to the formalisms adopted for representing domain and task knowledge. A domain ontology needs a formalism that allows to easily express taxonomical and non-taxonomical relationships among entities, which is static knowledge. A task ontology, instead, needs to represent dynamic operations like sequences, selections and iterations. The Semantic Web standard for representing ontologies is the Web Ontology Language [7]. While this is adequate for modelling domain knowledge, its lack of provision for dynamic operations lead us to add rules on top of OWL ontologies, represented using its standard candidate extension SWRL [8, 9]. One of the examples in which such an extension was necessary regarded an domain requiring meronomic relations [10], and Description Logic was not expressive enough to formalise them. Representing procedural knowledge, on the other hand, was accomplished mixing declarative rules with a traditional programming language (Java). Tasks then are represented by clauses (i.e. a set of premises in conjunction and a single consequence) whose consequence is represented by a block of code to be executed.

2.2 Annotation and search

The issue we are trying to solve is how to retrieve information from heterogeneous sources that are described w.r.t. an ontology that formalises the application domain. We divide the QuestSemantics framework into two stages, reflecting the two tasks of semi-automatic resource annotation and knowledge-based resource retrieval -the annotation stage and the search stage. In the first stage, we use both domain knowledge and task specific knowledge (e.g. layout specification, annotation and filter rules) in order to create semantic metadata about the information sources we want to exploit. This metadata is then used in the search stage, where specific queries from the application user are answered using the domain knowledge to guide the query process.

The Annotation stage is composed of four distinct process elements:

- Harvesting of live information sources, ensuring retrieved information is up to date with the latest information available.
- Analysis of the retrieved resources, using the knowledge encoded in the heuristic task rules, to identify which resources are of interest for the annotation component.
- Annotation of the analysis results using the domain ontologies - instances of concepts are identified, and, where possible, attributes are retrieved and relations between instances are stated.
- Storage of the metadata resulting from the annotation process in an RDF database.

The Search stage is primarily devoted to using a semantic query language, in this case the current W3C recommendation SPARQL [11], to retrieve specific information from the metadata stored in the last step of the Annotation phase. Queries will impose constraints upon potentially matching resources using the ontology representing the application domain. Responses to queries will consist of lists of matching resources, containing the metadata descriptions and pointer to the original source (e.g. web-page or database record-set). A graphical search interface enables user-specification of the semantic queries in an intuitive and non-technical manner, and allows clear presentation of and access to the resulting resources.

2.3 Design of the Framework

The framework design (depicted in Figure 1) is based around two software components: an Annotation Engine to analyse and filter the retrieved documents (handling the Annotation stage), and a semantic Search Engine to provide fine-grained access to the filtered documents (handling the Search stage). The two components also share a Store component, which is responsible for all data storage, consisting of document contents, ontologies and metadata instantiations, and the intermediate results created by the analysis and annotation components.

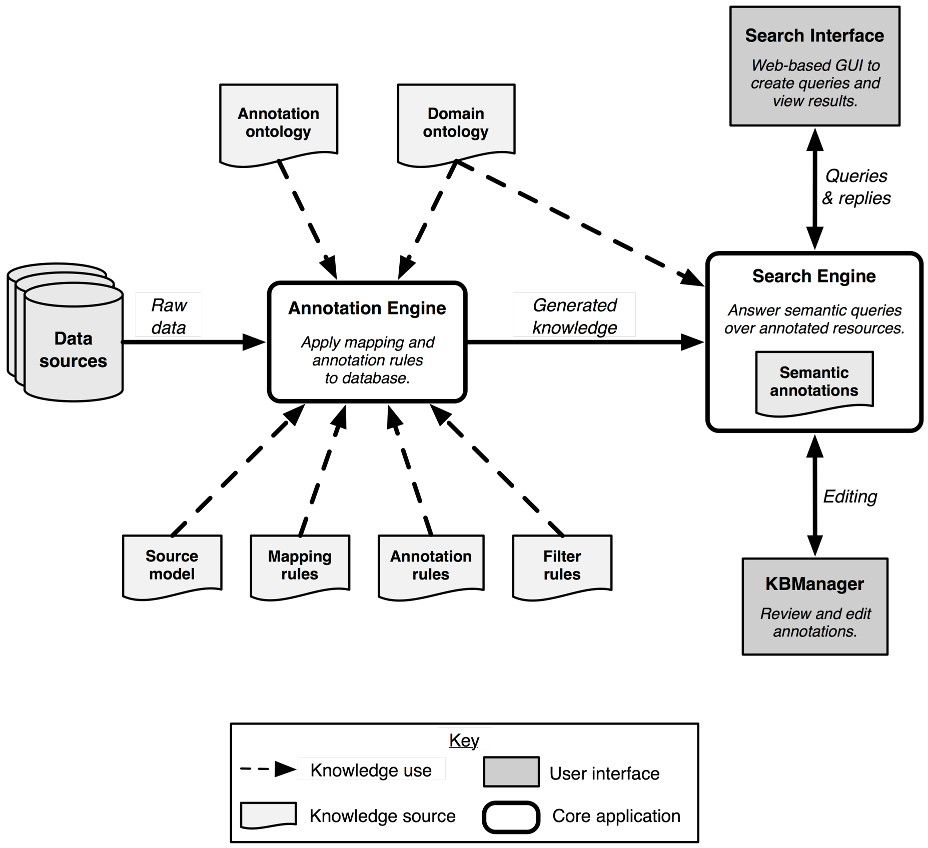

Fig. 1. General System Architecture

The Annotation Engine component retrieves documents from their sources, and then analyses, annotates and filters the documents on the basis of the application needs. Each of these functions is performed by a specific element, which is an implementation of one of the interfaces presented (Harvester, Analyzer, Annotation Engine, and Filter). Task specific knowledge is separated from domain knowledge at this level of abstraction: the Analyzer element is devoted to use the task specific knowledge available, e.g. how to find relevant information in a web page, while the Annotation Engine element uses domain knowledge in order to create the actual metadata. We obtained these independent components by leveraging the distinction between knowledge needed for each functionality, so that changes in task or domain only have impact on one component. Moreover, confining the task specific knowledge to the Analyzer system makes the Search component completely agnostic of the way information is retrieved, easing the process of using multiple knowledge-bases to answer users' queries. At the time of writing, two examples of this modular system have been implemented, which are presented in detail in Section 3 *Test-case Deployment and Evaluation*. The four elements of the Annotation Engine component are as follows.

Harvester element: An implementation of the Harvester interface must be able to retrieve information resources, and convert them into a form suitable for the annotation process. In the case of web pages, the Harvester retrieves the pages and saves them in the Store component as text documents. When the source is a database, as in one of the test cases we present later, it retrieves first the database schema and then the contents, and saves them in a XML format.

Analyzer element: Analyzer elements define methods to extract relevant information from an input information source, and store it in an intermediate format suitable for the Annotation Engine element. The architecture of this element is sketched in Fig. 2. Document layout specific information is encoded in the form of regular expressions (or with specialized Java code) into an implementation of the MatchingPattern interface. A set of these implementations is used by a Parser implementation, and a Parser together with its MatchingPattern elements forms a Rule. Rules are considered as atomic objects, meaning that the relevant information found by the MatchingPattern elements inside a Rule are only extracted if all the MatchingPattern are found to be satisfied in the input document/source; this is the case in which a Rule is said to be applicable. Some Rules can condition the applicability of other Rules, such as where one Rule determines that the current resource is unsuitable and forces all subsequent Rules to be skipped (a Blocking Rule), i.e. if Rule A is devoted to find an essential information, such as the reference number of the document that is being analyzed, when this Rule is not satisfied this means that the document is missing necessary information and is useless for search purposes (e.g. this can happen if an unrelated web page is being fed into the Analyzer element). When using the Analyzer over the XML structure extracted from a database, its only function is to translate from XML to the intermediate format.

Annotation element : This element creates the RDF models representing the information highlighted by the Analyzer - building source metadata according to the domain and application specific ontology(ies). Its internal architecture is sketched in Fig. 3. Analogously with the internal structure of the Analyzer element, annotation is performed by means of AbstractDocumentMatchingPattern implementations. Each imple-

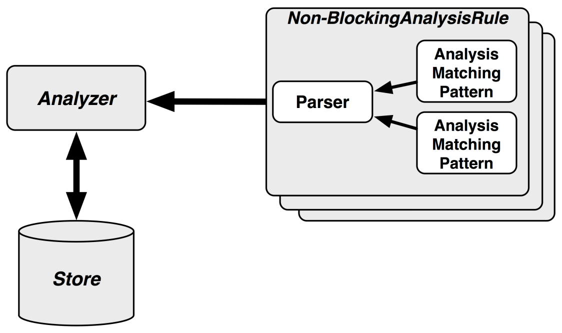

Fig. 2. Analyzer component detailed architecture

mentation extracts a specific piece of information from the Analyzer output, and Annotator processes create and formalise the metadata into an RDF model. Annotators and AbstractDocumentMatchingPatterns are grouped into AnnotationRules, which can be Blocking or Non-Blocking. The Annotation element is the first point in the process where the form of the source information becomes unimportant, i.e. it is agnostic w.r.t. whether the data is from web pages or from other sources, such as a database.

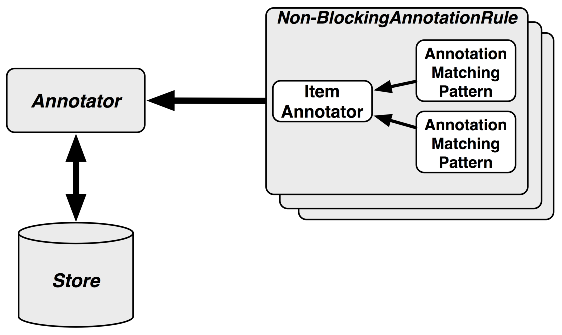
Non-BlockingAnnotationRule
Annotation Matching Pattern
Annotation Matching Pattern
Item Annotator
Annotator
Store

Fig. 3. Annotation component detailed architecture

Filter element: This element's role is to apply some pre-defined filter rules to determine whether a specific resource is suitable for use by the Search Engine. One example of such uses is the removal of information that is no longer up to date or useful (e.g. some information can expire after a certain amount of time, like a call for papers which is not useful after the submission date). It's architecture is outlined in Fig. 4.

Store element: Each step of the annotation process produces data that must be saved persistently, both for performance reasons (e.g. to save retrieved documents so that they

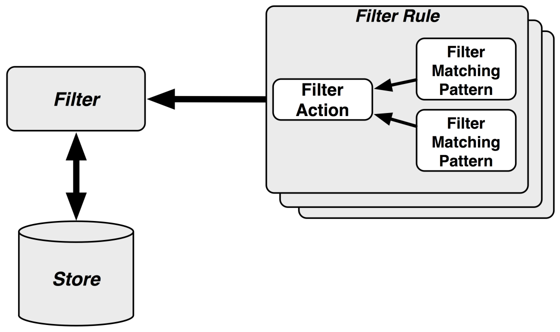

Fig. 4. Filter component detailed architecture

are available for the analysis step) and to keep track of connections between information items, such as the source of a specific annotation. The Store interface enables an application to save and retrieve data identified by a URI, such as byte streams (typically containing text documents such as HTML pages), Java maps containing intermediate mapping results, and RDF models containing finished annotations. In addition, the Store interface is designed to enable saving relations such as the fact that a specific URI is an alternate name for another resource, i.e., in OWL terms, the two resources are OWL:SAMEAS each other. This is particularly useful when a single conceptual resource is described by different documents and enables the annotation rules to retrieve all the available information for the resource, addressing the problem of information that is logically related but phisically disconnected. At the moment, it is possible to use either a file based persistence implementation (using files for byte streams, Java Properties files for Java maps and RDF/XML serialization for RDF models) or a database implementation, where all the information are placed in database tables (currently MySQL 5.0.x is being used for this task).

The Search Engine component of the framework is responsible for querying the information generated by the Annotation component. It stores the ontology, knowledge base, generated annotations and resource index, enabling access to the annotated documents on the basis of the encoded application knowledge. The search engine is intended to accept queries posed in SparQL, and will return a set of links to matching resources. This requires the search engine to first execute the SparQL against the stored RDF model, and then use the resulting instance URIs to obtain the relevant URLs from the document links data. It presents a specialised search interface, enabling users to develop an abstract model of a semantic query, pose it to the engine, and then review the resulting matched documents. The search interface provides the means by which end-users (i.e. people who are not experts in Semantic Web technologies) will access the resources filtered and annotated by the Annotation Engine component. It is also possible to add and delete entities and properties (with related values), so that a user can interact with the knowledge base to fine tune the query, enabling subsequent searches to become more accurate. There were two key aims for the query interface. First of all, we wanted the user to be presented with an intuitive and clear abstract query model, in order to hide as much as possible of the underlying complexity of representation and reasoning. As a second target we wanted to be able to use powerful semantic queries, that used as much as possible of the available information.

2.4 Tools and Languages

As already mentioned, the framework is built using Java version 5.0; the Semantic Web languages used are RDF[12] for metadata representation and OWL[7] as ontology language. The language used to query the knowledge base is SPARQL, which is an upcoming standard (Working Draft at the time of writing[1]). As intermediate format, XML[13] is used in both the Store component and as intermediate result for the database oriented implementation of Harvester; the libraries used in this case are those included in Java 1.5, which means the Xerces API for XML[2].

In order to access RDF models and ontologies, we have used the Jena Semantic Web Framework by HP[14], version 2.4; in order to build, refine and check the ontologies themselves, as well for reasoning and consistence checking purposes, we have used the Pellet DL reasoner[3] and the SWOOP ontology editor[4]. As for the SPARQL language, the implementation we use is the ARQ engine which is developed by the Jena team and included in the Jena download.

In order to develop the client side of the Search Engine in the easiest way, we used GWT (Google Web Toolkit[5]) in order to have an easy way to code Javascript interfaces through Java code and produce a AJAX compliant application; the use of the toolkit enables us to build cross-browser applications that do not overload the server with page reload requests in a very easy way.

The database we use for storage, as said earlier, is MySQL[6] version 5.0.x, through its JDBC driver. The Harvester implementation has been used to access both MySQL databases and SQL Server[7] databases. Our server side application server is Tomcat[8] version 5.5.x, which complies with JSP specifications 2.0[9] / Servlet specifications 2.4[10].

[1] http://www.w3.org/TR/rdf-sparql-query/
[2] http://xerces.apache.org/
[3] http://pellet.owldl.com/
[4] http://www.mindswap.org/2004/SWOOP/
[5] http://code.google.com/webtoolkit/
[6] www.mysql.com
[7] http://www.microsoft.com/sql/default.mspx
[8] http://tomcat.apache.org/
[9] http://java.sun.com/products/jsp/
[10] http://java.sun.com/products/servlet/reference/api/index.html

3 Test-case Deployment and Evaluation

The QuestSemantics system has been deployed in two different commercial test-cases. The first commercial partner was Vectra Group Ltd. Their problem was one of information overload - they need to examine specific web-published documents for commercial opportunities matching their areas of business interest. However, their current search service only uses keywords to represent these interests and match against the publications - resulting in many potential matches, which then need to be human-filtered to determine if they represent suitable commercial opportunities. QuestSemantics was applied to this task of information retrieval, to enable more domain-specific analysis and filtering of the published documents. The knowledge representation formalisms are used to encode knowledge about the areas of business in which they are interested (i.e. sectors, markets, activities, companies, locations, etc.), and knowledge about the source material regarding how to find, annotate and filter those sources on the basis of the business knowledge. The application runs a daily, automatic annotation and filtering process of potentially matching resources, storing the results in the knowledge-base. This metadata is then accessed via the search interface to perform regular searches over sub-sets of the company's business interests for suitable opportunities.

Application of QuestSemantics to this task gives more accurate results from the resource matching process, producing fewer false-positive matches for the business criteria. This allows Vectra to concentrate efforts on a more precise set of results, reducing the time spent checking which of the possible matches are actual matches. The increased result accuracy also aids identification of suitable resources, that may be over-looked in the current process due to information overload. In addition, providing fine-grained access to potentially matching resources through an advanced search interface enables Vectra to perform on-demand search, on the basis of the business knowledge, for resources matching specific criteria rather than having to determine this by a manual search of all resources.

The second commercial test-case concerns knowledge-based search over pre-existing database information resources. The North West Aerospace Association (NWAA) maintains a database of its member aerospace companies, giving details of these companies (areas of expertise, specific capabilities, etc.). Access to this database is provided through the NWAA website, enabling interested parties to search for aerospace companies. However, the current search is inflexible, with only a basic categorization of activities, capabilities and approvals, and cannot combine search features - meaning that searches can only be approximate and do not allow identification of companies exhibiting specific feature combinations without manual cross-referencing of search results.

The application of QuestSemantics enables creation of a knowledge-base, based on an ontology of the domain, using the company data currently held in NWAA's database, and provides a semantic search facility allowing the knowledge-base to be searched by constructing specific queries based upon the ontological model. The knowledge represented in the ontology is a conceptualistion of the aerospace domain in terms of the features, capabilities and business relationships applying to companies within that domain. This conceptualisation is then instanciated to describe the specific companies and ancillary information, gathered from existing database resources. The annotation rules, layered on top of the ontology, specify how the existing information is automatically mapped into this knowledge representation. The knowledge-base is thus dynamically created from the existing data resources, and is updated on demand. The search interface to this knowledge-base is designed to be used through the NWAA website by companies seeking partners with specific aerospace expertise. The semantic search enables use of multiple, hierarchically structured categorisations and features, combination of features using boolean logic, aggregation of results over similar categories, and reference to specific company features within search constraints. The primary benefits of the enhanced search facility to NWAA and its members are more accurate results for all types of search over company information, leading to a saving of company time analysing search results in order to identify potential partner companies.

We have performed an evaluation of the performance of the annotation and filtering system when applied to Vectra's problem of identifying web resources that match business interests. The evaluation examined a large-scale harvest of 34285 documents, determining how many are returned by the QuestSemantics system, and, of these, how many are of genuine business interest to Vectra. These results (shown in Table 1) demonstrate that only a very small fraction of the published documents are of genuine interest to Vectra, which matches with their expectation. Furthermore, the results show that the semantic annotation and filtering process is performing well, eliminating over 93 percent of published documents with a British location (GB). The results for QuestSemantics compare well with the results for the current service. A full comparative evaluation is still ongoing in this regard, however, random spot-check comparisions over individual daily returns show an average reduction in returns of 71 percent. The effects on Vectra's business have been significant - ceasing subscription to the existing search service, and now intending to use QuestSemantics. However, there is still significant room for improvement on the current results, as only 3.5 percent of the returns from QuestSemantics were determined to be of genuine business interest to Vectra. The main targets for these improvements are those documents were a search term has syntactically matched, but has not fully matched the intention of the search term. This can be addressed by further application of natural language understanding techniques, taking fuller account of context and grammar.

	Contracts	**GB contracts**	**Returned**	**Interesting**
Total	34285	2894	199	7
Daily average	836.22	70.59	4.83	0.17

Table 1. Summary of Vectra test-case evaluation.

Development and Maintenance The development of the QuestSemantics software was undertaken in parallel with it's application to the two test-cases. Both of the projects had the following development milestones:

- *Knowledge modeling and ontology construction*: by elicitation of knowledge from domain experts and analysis of existing data, etc.
- *Development of annotation engine*: to parse, analyse and filter resources, and to create resource meta-data.
- *Development of search interface*: to enable knowledge-based access to annotated resources.
- *Deployment of integrated system*: onto user systems, followed by evaluation and user-feedback.

However, there was a degree of overlap between these tasks in order to ensure a smooth development progression. There were four people involved in the development of the QuestSemantics application, and the effort expended on the different aspects of the two test-case development projects is show in Table 2. As can be seen from the table, the distribution of effort is significantly different in the two test-cases. The majority of the software design and development occurs in the Vectra project largely because this was the first project and much of the development here was intentionally reused in the second project. However, more effort was expended on knowledge engineering in the NWAA project due to the fact that the domain was more complex to model, requiring additional modelling of concept relationships (e.g. meronomy).

	Vectra	**NWAA**
Knowledge elicitation and modeling	4 pm	8 pm
Software design, development and testing	11 pm	7 pm
Deployment and evaluation	4 pm	2 pm

Table 2. Summary of effort in person/months in test-cases.

The deployments of QuestSemantics have been performed as a one-off, test-case deployments, onto the commerical partners' own systems as local applications. Whilst we have little maintenance experience as yet, maintenance updates to the systems are expected. Some of the expected maintenance activities are addressed providing the partners with the ability to amend and update the knowledge-base themselves, through a specially developed tool. Therefore the company maintains the knowledge used by the application, which will only change in-line with their changing business interests, and so is not very dynamic. Additional maintenance activities will be performed by the developers through continued development and update of the tool, in addition to responding to any software problems encountered. This intended maintenance is underpinned by the modular design of the software, and by the strict separation of business knowledge and software functionality.

There are many ways in which the current application could be improved, both in the existing tasks of annotation and search, and in extensions to the current system to address areas such as knowledge management and business intelligence. Examples of such improvements for the Vectra test-case are:

- Extension of filter rules to consider specific rule-exceptions, thus allowing more flexible application of filters.
- Refinement of the query construction and editing methodology, enabling a more intuitive and flexible workflow.
- Search result ordering can be extended to allow a variety of rankings, based on different criteria, to be applied.
- Annotation lifecycle management can be enhanced to revise and remove annotations describing resources in a fully automated manner.
- Allow users to add further annotations to retrieved resources, indicating what action is being taken, which would then enable monitoring of activity in this domain.
- Extensions to the knowledge-base regarding closely related business areas would enable monitoring of opportunities on the margins of current business interests - helping to identify areas of potential business expansion.

4 Related Work

Here we survey some of the relevant existing approaches for annotating and searching web resources, based on Semantic Web technologies. One of the the first semantic annotation applications was Annotea [15] in 2001. Annotea employs RDF Schema as formalism to express the meta-data vocabulary, but the resource annotation process is entirely manual. A manual annotation process tends to be subjective (i.e. depends on the knowledge and point of view of the domain expert), and is time-consuming and tedious - leading to a second generation of semi-automatic semantic annotation tools. Besides automatizing parts of the process, such tools also proposed a slightly more constraining notion of annotation - it shifted from being generic information related to (a portion of) a document, to being a formal description of the information within it. In [16] the authors present a platform (Seeker), and an application (SemTag) built upon it, that were designed to scale up to annotation for the whole web. SemTag relies on a fixed ontology, namely TAP [17], as its meta-data vocabulary, and identifies instances of the concepts appearing in the TAP ontology within the analysed documents. This is accomplished by means of an algorithm for word sense disambiguation that considers word windows around a term as context to help in disambiguating its sense.

The S-CREAM [18] abstract framework, and its implementation Ont-O-Mat, represent an evolution of such approaches. They do not depend on the use of a particular ontology and individuate instances, relations between instances, and instance attributes (relationships between instances and values). They employ Amilcare [19], a tool for learning adaptive rules for tagging a corpus that leverages several Natural Language Processing methodologies. The gap between XML-based Amilcare annotation and Semantic Web meta-data formalised w.r.t. an ontology is bridged, within the S-CREAM architecture, by a Discourse Representation component, which is responsible for translation from Amilcare results into the meta-data in a Semantic Web standard language.

The more recent Knowledge Parser [20] proposes an architecture that explicitly accounts for layout processing as one of the early steps in the annotation process. Annotations can be based on multiple ontologies that are not known a priori. Knowledge Parser has a separate process (Intelligent Ontology Population) for the creation of instances of the concepts in the ontologies. This process varies according to the domain, in that the rules (policies) for populating the ontology are dependent on the application. It provides a Natural Language interface for querying the generated knowledge base.

Knowledge Parser is the only one of the above systems that provides a search interface, and only the latter two are domain independent systems. S-CREAM, Knowledge Parser and our system can be categorised as systems that aim to employ semantic annotation and access in very specific knowledge domains. On the contrary, SemTag and analogous systems (e.g. the KIM platform [21]) have been designed for *bootstrapping* the Semantic Web by annotating the current Web, and rely on very general ontologies in order to capture the widest possible range of knowledge. Therefore, although the SemTag-like category of tools need little customisation in order to be used in any domain, they cannot be easily adapted to produce very detailed annotation regarding specific domains.

5 Conclusions

As can be seen from the evaluation in the Vectra use-case (see Section 3), the application of knowledge representation methodologies to intelligent data capture and access can produce very successful results. The significant reduction in false positive returns produces savings in company time and effort expended to identify opportunities, and helps to reduce the likelyhood that suitable opportunities are missed due to information overload. In addition, as shown with both Vectra and NWAA, the facility to access the data resources on the basis of the encoded business knowledge enables users to identify useful resources in a way that is tailored to their needs and experience. Furthermore, our focus upon limited and clearly defined domains of knowledge enables the business partners to specify the conceptualisation needed to apply their implicit knowledge about their business to the problem tasks in an automated manner.

As a result of the two test-case applications of the QuestSemantics system, a number of lessons have been learnt regarding the application of knowledge representation and manipulation techniques within commercial scenarios. Companies require end-to-end solutions that solve specific business problems, requiring development of an integrated system of knowledge representation and other technologies to solve the whole of that problem. The knowledge elicitation process requires significant time and effort, but, in our experience, the rich expressivity of the formalisms employed provides a straightforward means to encode the knowledge required. To enable the business partners to make full use of the application, the presentation of the knowledge is as important as its representation. The languages employed provide assistance here by allowing concepts, properties and values to be represented in a natural way that supports an expressive but clear presentation. Finally, our focus upon maintaining a strict separation between the various different types of knowledge represented, both problem-specific and generic, underpins the flexiblility of our approach, and enables its application to almost any domain, given a sufficiently detailed ontology and annotation rules.

References

1. Studer, R., Benjamins, R., Fensel, D.: Knowledge engineering: Principles and methods. Journal of the ACM **25**(1-2) (March 1998) 161–197
2. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Journal of Web Semantics **4**(1) (2006)
3. Miller, G.: Wordnet: a lexical database for english. Communications of the ACM **38**(11) (November 1995) 39–41
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001)
5. van Heijst, G., Schreiber, A.T., Wielinga, B.J.: Using explicit ontologies in kbs development. Int. J. Hum.-Comput. Stud. **46**(2) (1997) 183–292
6. Bylander, T., Chandrasekaran, B.: The right level of knowledge acquisition. In: Knowledge acquisition for knowledge based systems. Volume 1. Academic Press (1988)
7. McGuinness, D.L., van Harmelen (Eds), F.: OWL Web Ontology Language Overview (2004) http://www.w3.org/TR/owl-features/.
8. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004) http://www.w3.org/Submission/SWRL/.
9. Horrocks, I., Patel-Schneider, P.F., Bechhofer, S., Tsarkov, D.: OWL rules: A proposal and prototype implementation. J. of Web Semantics **3**(1) (2005) 23–40
10. Winston, M.E., Chaffin, R., Herrmann, D.: A taxonomy of part-whole relations. Cognitive Science **11**(4) (1987) 417–444
11. SPARQL: Query language for RDF. W3C Candidate Recommendation - 14th June 2007
12. : RDF model and syntax specification (2004) http://www.w3.org/TR/REC-rdf-syntax.
13. : Extensible Markup Language (XML) 1.0 (Fourth Edition) (2006) http://www.w3.org/TR/2006/REC-xml-20060816/.
14. McBride, B.: JENA: A Semantic Web toolkit. IEEE Internet Computing **6** (Nov-Dec 2002) 55–59
15. Kahan, J., Koivunen, M.R.: Annotea: an open RDF infrastructure for shared Web annotations. In: Proceedings of the 10th International Conference on the World Wide Web. (2001) 623–632
16. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: SemTag and Seeker: bootstrapping the Semantic Web via automated semantic annotation. In: Proceedings of the 12th International Conference on the World Wide Web. (2003) 178–186
17. Guha, R.V., McCool, R.: TAP: A semantic web test-bed. Journal of Web Semantics **1**(1) (2003) 81–87

18. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM - Semi-automatic CREAtion of Metadata. In: Proceedings of Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference. (2002) 358–372
19. Ciravegna, F., Dingli, A., Wilks, Y., Petrelli, D.: Timely and non-intrusive active document annotation via adaptive information extraction. In: Proceedings of the ECAI Workshop on Semantic Authoring, Annotation and Knowledge Markup., Lyon, France. (2002)
20. Rodrigo, L., Benjamins, V.R., Contreras, J., Patón, D., Navarro, D., Salla, R., Blázquez, M., Tena, P., Martos, I.: A Semantic Search Engine for the International Relation Sector. In: Proceedings of the 4th International Semantic Web Conference. (2005) 1002–1015
21. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. Journal of Web Semantics **2**(1) (2004) 49–79