ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008
© 2008 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-58603-891-5-648

QuestSemantics - Intelligent Search and Retrieval of Business Knowledge

Ian Blacoe and **Ignazio Palmisano** and **Valentina Tamma**[1] and **Luigi Iannone**[2]

Abstract. Keyword-based search engines, though hugely popular, show limitations when trying to answer very specific queries. The processing of search results is performed by users, rather than by software. *Ontologies* provide a means to create formal, machine-processable descriptions of the knowledge in a certain domain [14], and, by using elements of these descriptions to annotate suitable information sources, they can be analysed and manipulated in an *intelligent* manner. The QuestSemantics platform provides automated ontology-based metadata creation and resource annotation, with subsequent ontology-based querying of the annotated resources. The 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 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 [15]. 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 with company documents and databases. Current keyword based search engines present limitations in that they cannot fully capture the richness intrinsic in natural language; e.g., synonymy and polysemy pose hard to solve problems for keyword based search task. Enhancing search engines with lexicons such as WordNet [11] can help to relieve these problems, but it 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 technologies. The Semantic Web [1] is an evolution of the current Web where information is represented in a machine-readable format, while maintaining the human-friendly HTML representation. Ontologies [14] are crucial in providing shared and machine processable *meaning* to web resources. An ontology models the entities and processes that are used to describe both the content of a web resource, and, more importantly, the logical relations between the resources. 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. The annotation process normally involves the creation of 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.

This paper presents QuestSemantics (QS), 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. QS 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 includes 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.

The paper illustrates two different commercial use-cases in which the QS 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 this paper is organised as follows. Section 2 describes the design and implementation of the developed application. Section 3 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. Related work is described in Section 4, and in Section 5 some conclusions are drawn.

2 DESIGN AND IMPLEMENTATION

QS 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 differ only slightly between different applications and different domains. The aim of the 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 remainder of this section the main aspects underlying the design and implementation of QS are discussed.

2.1 Knowledge independent components

QS is a generic platform for automatic annotation of semi-structured information sources and retrieval based on semantic queries (i.e.

[1] Computer Science Department, University of Liverpool; email: {I.W.Blacoe, I.Palmisano, V.Tamma}@liverpool.ac.uk
[2] Computer Science Department, University of Manchester; email: iannone@cs.man.ac.uk

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 is limited to domain related aspects only. In QS design there is 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 [16], and thus describes the ways to perform useful changes to the domain states.

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 the easy expression of taxonomical and non-taxonomical relationships among entities, i.e. 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 (OWL) [10]. While this is adequate for modelling domain knowledge, it is not suitable to represent dynamic operations; therefore rules are added on top of OWL ontologies, and they are represented using SWRL [7, 6]. One of the examples in which such an extension was necessary regarded the expression of meronomic relations [17]; Description Logic is not expressive enough to formalise this. Representing procedural knowledge, on the other hand, is accomplished mixing declarative rules with a traditional programming language (Java). Tasks are then represented by clauses, i.e. a set of premises in conjunction and a single consequence, and the consequence is represented by a block of executable code.

2.2 Annotation and Search

The framework is divided 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, both domain knowledge and task specific knowledge (e.g. layout specification, annotation and filter rules) are used in order to create semantic metadata about the information sources to exploit. This metadata is then used in the search stage, where specific queries from the 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 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[3] database.

The Search stage is primarily devoted to retrieve specific information from the metadata stored in the last step of the Annotation phase. Queries are expressed in SPARQL [13], and will impose constraints upon potentially matching resources using the ontology representing the application domain. Responses to queries will be lists of matching resources, containing the metadata descriptions and a 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.

[3] http://www.w3.org/RDF/

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. The Annotation Engine component retrieves documents

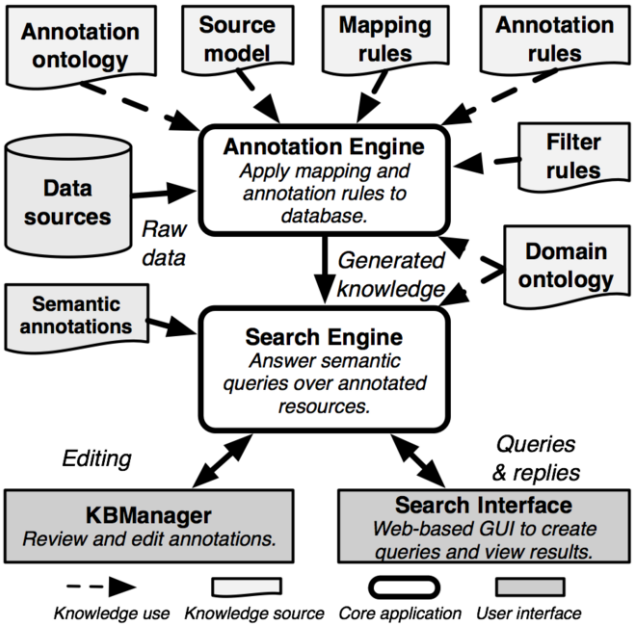

Figure 1. General System Architecture

from their sources, and then analyses, annotates and filters them on the basis of the application needs. Each of these functions is performed by a specific element, that is an implementation of one of the interfaces presented (Harvester, Analyzer, Semantic Annotator). Task specific knowledge is separated from domain knowledge at this level of abstraction: the Analyzer element oses only the task specific knowledge available, e.g. how to find relevant information in a web page, while the Semantic Annotator element uses domain knowledge in order to create the actual metadata. These independent components are obtained by leveraging the distinction between the knowledge needed for each functionality, so that changes in task or domain only have an impact on one component. Moreover, confining the task specific knowledge to the Analyzer system makes the Search component completely agnostic to 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, presented in detail in Section 3, have been implemented. The 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 presented later, it retrieves first the database schema and then the contents, and saves them in an 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. Its architecture is shown in Figure 2. Document layout specific information is encoded in the form of regular expressions (or with specialised 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, e.g., one Rule determines that the current resource is unsuitable and forces all subsequent Rules to be skipped (a Blocking Rule).

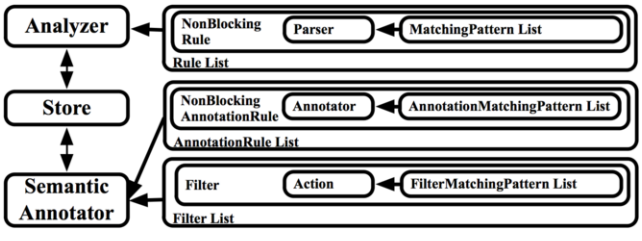

Figure 2. Annotation components detailed architecture

Semantic Annotator 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 architecture is shown in Figure 2. Analogously with the internal structure of the Analyzer element, annotation is performed by means of AbstractDocumentMatchingPattern implementations. Each implementation 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 Semantic Annotator 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 originate from web pages or from other sources, such as a database. Filters in the Semantic Annotator are used to apply some predefined 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). Details are shown in Figure 2.

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 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. 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 physically disconnected.

The Search Engine component of the framework is responsible for querying the information generated by the Annotation component; it is intended to accept queries posed in SPARQL, and will return a set of links to matching resources. A specialised search interface enables the 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 Semantic Annotator 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. The key aim for the query interface is that the user has 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.

3 DEPLOYMENT AND EVALUATION

The QS system has been deployed in two different commercial testcases. The first commercial partner is 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. QS 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 meta-data is then accessed via the search interface to perform regular searches over sub-sets of the company's business interests for suitable opportunities.

The application of QS to this task enables the resource matching process to obtain more accurate results, 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 infor-

mation 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 web site, enabling interested parties to search for aerospace companies. However, the current search is inflexible, with only a basic categorisation of activities, capabilities and approvals, and cannot combine search features. This means 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 QS enables the 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 conceptualisation of the aerospace domain in terms of the features, capabilities and business relationships applying to companies within that domain. This conceptualisation is then instantiated 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 web site by companies seeking partners with specific aerospace expertise. The semantic search enables the 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 spent analysing search results in order to identify potential partner companies.

In the Vectra LTD use case, an evaluation of the performance of the annotation and filtering system, applied to the problem of identifying web resources that match business interests, has been performed. The evaluation examined a large-scale harvest of 34285 documents, determining how many are returned by the QS 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 % of published documents with a British location (GB). The results for QS compare well with the results of the current service. A full comparative evaluation is still ongoing in this regard, however, random spot-check comparisons over individual daily returns show an average reduction in returns of 71 %. The effects on Vectra's business have been significant; Vectra has ceased subscription to the existing search service, and now intends to use QS. However, there is still significant room for improvement on the current results, as only 3.5 % of the returns from QS were determined to be of genuine business interest to Vectra.

	Contracts	GB contracts	Found	Interesting
Total	34285	2894	199	7
Daily avg	836.22	70.59	4.83	0.17

Table 1. Summary of Vectra test-case evaluation.

There are many ways in which the current application could be improved, both in the existing tasks of annotation and search, and in the 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

This section presents a brief survey 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 [8] in 2001. Annotea employs RDF Schema as its 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. These aspects lead to a second generation of semi-automatic semantic annotation tools. Besides automatising parts of the process, these tools also propose a slightly more constrained notion of annotation. An annotation shifted from being generic information related to (a portion of) a document, to being a formal description of the information within it. In [3] 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 [4], 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 [5] 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 [2], 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 [12] 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 systems reviewed that provides a search interface, and only the latter two are domain independent systems. S-CREAM, Knowledge Parser and QS 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 [9]) 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 on identifying opportunities, and helps to reduce the likelihood 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, 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 QS 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 mean to encode the knowledge required; the main problem identified in this phase is the need to confront business managers with the formalized knowledge in order to validate it; this process requires a basic understanding of the involved technologies, which can require a relevant effort in terms of company time. Therefore, 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, the strict separation between the various different types of knowledge represented, both problem-specific and generic, underpins the flexibility of the approach, and enables its application to almost any domain, given a sufficiently detailed ontology and annotation rules.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila, 'The Semantic Web', *Scientific American*, (May 2001).
[2] F. Ciravegna, A. Dingli, Y. Wilks, and D. Petrelli, 'Timely and non-intrusive active document annotation via adaptive information extraction', in *Proc. of the ECAI Workshop on Semantic Authoring, Annotation and Knowledge Markup.*, Lyon, France., (2002).
[3] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien, 'SemTag and Seeker: bootstrapping the Semantic Web via automated semantic annotation', in *Proc. of the 12th International Conference on the World Wide Web*, pp. 178–186, (2003).
[4] R.V. Guha and R. McCool, 'TAP: A semantic web test-bed.', *Journal of Web Semantics*, **1**(1), 81–87, (2003).
[5] S. Handschuh, S. Staab, and F. Ciravegna, 'S-CREAM - Semi-automatic CREAtion of Metadata.', in *Proc. of Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference*, pp. 358–372, (2002).
[6] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov, 'OWL rules: A proposal and prototype implementation', *J. of Web Semantics*, **3**(1), 23–40, (2005).
[7] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, 2004. http://www.w3.org/Submission/SWRL/.
[8] J. Kahan and M.R. Koivunen, 'Annotea: an open RDF infrastructure for shared Web annotations.', in *Proc. of the 10th International Conference on the World Wide Web*, pp. 623–632, (2001).
[9] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, 'Semantic annotation, indexing, and retrieval', *Journal of Web Semantics*, **2**(1), 49–79, (2004).
[10] D.L. McGuinness and F. van Harmelen (Eds). OWL Web Ontology Language Overview, 2004. http://www.w3.org/TR/owl-features/.
[11] G.A. Miller, 'Wordnet: a lexical database for english.', *Communications of the ACM*, **38**(11), 39–41, (November 1995).
[12] L. Rodrigo, V.R. Benjamins, J. Contreras, D. Patón, D. Navarro, R. Salla, M. Blázquez, P. Tena, and I. Martos, 'A Semantic Search Engine for the International Relation Sector.', in *Proc. of the 4th International Semantic Web Conference*, pp. 1002–1015, (2005).
[13] SPARQL. Query language for RDF. W3C Recommendation, 15th January 2008.
[14] R. Studer, R. Benjamins, and D. Fensel, 'Knowledge engineering: Principles and methods.', *Journal of the ACM*, **25**(1-2), 161–197, (March 1998).
[15] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, 'Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art', *Journal of Web Semantics*, **4**(1), (2006).
[16] G. van Heijst, A.Th. Schreiber, and B.J. Wielinga, 'Using explicit ontologies in kbs development.', *Int. J. Hum.-Comput. Stud.*, **46**(2), 183–292, (1997).
[17] M.E. Winston, R. Chaffin, and D. Herrmann, 'A taxonomy of part-whole relations.', *Cognitive Science*, **11**(4), 417–444, (1987).