Available online at www.sciencedirect.com
ScienceDirect
Web Semantics: Science, Services and Agents on the World Wide Web 5 (2007) 72–105
JOURNAL OF Web Semantics
www.elsevier.com/locate/websem

AquaLog: An ontology-driven question answering system for organizational semantic intranets

Vanessa Lopez*, Victoria Uren, Enrico Motta, Michele Pasin

Knowledge Media Institute & Centre for Research in Computing, The Open University, Walton Hall, Milton Keynes MK7 6AA, United Kingdom

Received 29 July 2005; accepted 23 March 2007
Available online 31 March 2007

Abstract

The *semantic web* vision is one in which rich, *ontology-based semantic markup* will become widely available. The availability of semantic markup on the web opens the way to novel, sophisticated forms of question answering. AquaLog is a portable question-answering system which takes queries expressed in natural language and an ontology as input, and returns answers drawn from one or more knowledge bases (KBs). We say that AquaLog is portable because the configuration time required to customize the system for a particular ontology is negligible. AquaLog presents an elegant solution in which different strategies are combined together in a novel way. It makes use of the GATE NLP platform, string metric algorithms, WordNet and a novel ontology-based *relation similarity service* to make sense of user queries with respect to the target KB. Moreover it also includes a learning component, which ensures that the performance of the system improves over the time, in response to the particular community jargon used by end users.

Keywords: Semantic web; Question answering; Ontologies; Natural language

1. Introduction

The *semantic web* vision [7] is one in which rich, *ontology-based semantic markup* is widely available, enabling an author to improve content by adding meta-information. This means that with structured or semi-structured documents, texts can be semantically marked-up and ontological support for term definition provided, both to enable sophisticated interoperability among agents, e.g., in the e-commerce area, and to support human web users in locating and making sense of information. For instance, tools such as Magpie [19] support semantic web browsing by allowing users to select a particular ontology and use it as a kind of 'semantic lens', which assists them in making sense of the information they are looking at. As discussed by McGuinness in her essay on "Question Answering on the Semantic Web" [41], the availability of semantic markup on the web also opens the way to novel, sophisticated forms of question answering, which not only can potentially provide increased precision and recall compared to today's search engines, but are also capable of offering additional functionalities, such as (i) proactively offering additional information about an answer, (ii) providing measures of reliability and trust and/or (iii) explaining how the answer was derived. Therefore, most emphasis in QA is currently on the use of ontologies on the fly to mark-up text and make its retrieval smarter by using query expansion [41].

While semantic information can be used in several different ways to improve question answering, an important (and fairly obvious) consequence of the availability of semantic markup on the web is that this can indeed be queried directly. In other words, we can exploit the availability of semantic statements to provide precise answers to complex queries, allowing the use of inference and object manipulation. Moreover, as semantic markup becomes ubiquitous, it will become advantageous to be able to ask queries and obtain answers, using natural language (NL) expressions, rather than the keyword-based retrieval mechanisms used by the current search engines.

For instance, we are currently augmenting our departmental web site in the Knowledge Media Institute (KMi), http://kmi.open.ac.uk, with semantic markup, by instantiating an ontology

* Corresponding author. Fax: +44 1908 653169.
E-mail addresses: v.lopez@open.ac.uk (V. Lopez), v.s.uren@open.ac.uk (V. Uren), e.motta@open.ac.uk (E. Motta), m.pasin@open.ac.uk (M. Pasin).

1570-8268/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2007.03.003

describing academic life [1] with information about our personnel, projects, technologies, events, etc., which is automatically extracted from departmental databases and unstructured web pages. In the context of standard, keyword-based search this semantic markup makes it possible to ensure that standard search queries, such as "Peter Scott home page KMi", actually return Dr Peter Scott's home page as their first result, rather than some other resource (as is the case when using current non-semantic search engines on this particular query). Moreover, as pointed out above, we can also query this semantic markup directly. For instance, we can ask a query such as "which are the projects in KMi related to the semantic web area" and, thanks to an inference engine able to reason about the semantic markup and draw inferences from axioms in the ontology, we can then get the correct answer.

This scenario is of course very similar to asking natural language queries to databases (NLIDB), which has long been an area of research in the artificial intelligence and database communities [8,30,2,10,28], even if in the past decade it has somewhat gone out of fashion [3,27]. However, it is our view that the semantic web provides a new and potentially very important context in which results from this area of research can be applied. Moreover, interestingly from a research point of view, it provides a new 'twist' on the old issues associated with NLDB research.

As pointed out in Ref. [24] the key limitation of the NL interfaces to databases is that it presumes the knowledge the system is using to answer the question is a structured knowledge base in a limited domain. However, an important advantage is that a knowledge based question answering system can help with answering questions requiring situation-specific answers, where multiple pieces of information need to be combined and therefore the answers are being inferred at run time, rather than reciting a pre-written paragraph of text [12].

Hence, in the first instance, the work on the AquaLog query answering system described in this paper is based on the premise that the semantic web will benefit from the availability of natural language query interfaces, which allow users to query semantic markup viewed as a structured knowledge base. Moreover, similarly to the approach we have adopted in the Magpie system, we believe that in the semantic web scenario it makes sense to provide query answering systems on the semantic web, *which are portable with respect to ontologies*. In other words, just as in the case of tools such as Magpie, where the user is able to select an ontology (essentially a semantic viewpoint) and then browse the web through this semantic filter, our AquaLog system allows the user to choose an ontology and then ask queries with respect to the universe of discourse covered by the ontology.

Hence, AquaLog, our natural language front-end for the semantic web, is a portable question-answering system which takes queries expressed in natural language and an ontology as input, and returns answers drawn from one or more knowledge bases (KBs), which instantiate the input ontology with domain-specific information. Thus, AquaLog is especially suitable as front end to organizational semantic intranets where an organizational ontology is used as the basics for semantic mark up. Of relevance, is the valuable work done in AquaLog on the syntactic and semantic analysis of the questions. AquaLog makes use of the GATE NLP platform, string metric algorithms, WordNet and novel ontology-based *similarity services for relations and classes* to make sense of user queries with respect to the target knowledge base. Also, AquaLog is coupled with a portable and contextualized learning mechanism to obtain domain-dependent knowledge by creating a lexicon. The learning component ensures that the performance of the system improves over time, in response to the particular community jargon of the end users.

Finally, although AquaLog has primarily been designed for use with semantic web languages, it makes use of a generic plug-in mechanism, which means it can be easily interfaced to different ontology servers and knowledge representation platforms.

The paper is organized as follow: in Section 2 we present an example of AquaLog in action. In Section 3 we describe the AquaLog architecture. In Section 4 we describe the Linguistic Component embedded in AquaLog. In Section 5 the novel Relation Similarity Service. In Section 6 the Learning Mechanism. In Section 7 the evaluation scenario, followed by discussion and directions for future work in Section 8. In Section 9 we summarize related work. In Section 10 we give a summary. Finally, an appendix is attached with examples of NL queries and their equivalent triple representation.

2. AquaLog in action: illustrative example

First of all, at a coarse-grained level of abstraction, the AquaLog architecture can be characterized as a cascaded model, in which a NL query gets translated by the Linguistic Component into a set of intermediate triple-based representations, which are referred to as the Query-Triples. Then, the Relation Similarity Service (RSS) component takes as an input these Query-Triples and further processes them to produce the ontology-compliant queries, called Onto-Triples, as shown in Fig. 1.

The data model is triple-based, namely it takes the form of ⟨subject, predicate, object⟩. There are two main reasons for adopting a triple-based data model. First of all, as pointed out by Katz et al. [31], although not all possible queries can be represented in the binary relational model, in practice these exceptions occur very infrequently. Secondly, RDF-based knowledge representation (KR) formalisms for the semantic web, such as RDF itself [47] or OWL [40] also subscribe to this binary relational model and express statements as ⟨subject, predicate, object⟩. Hence, it makes sense for a query system targeted at the semantic web to adopt a triple-based model that shares the same format as many millions of other triples on the Semantic Web.

For example, in the context of the academic domain in our department, AquaLog is able to translate the question "what is the homepage of Peter who has an interest on the semantic web?" into the following, ontology-compliant logical query, ⟨what is?, has-web-address, peter-scott⟩ and ⟨person?, has-research-interest, Semantic Web area⟩, expressed as a conjunction of non-ground triples (i.e., triples containing variables).

In particular, consider the query "what is the homepage of Peter?", the role of the RSS is to map the intermediate form,

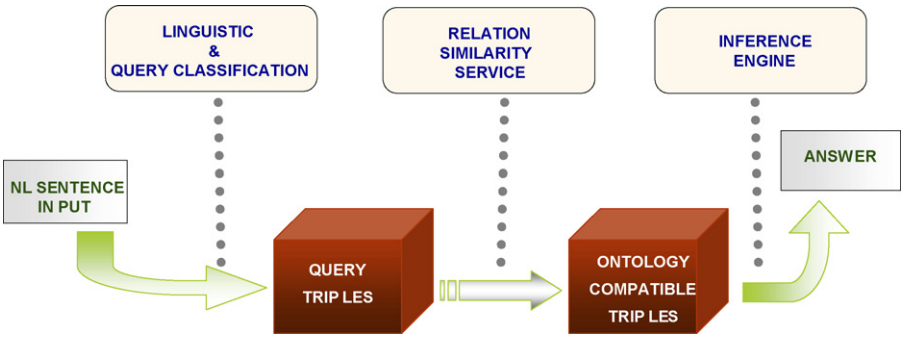

Fig. 1. The AquaLog data model.

⟨what is?, homepage, peter⟩, obtained by the linguistic component, into the target, ontology-compliant query.

The RSS calls the user to participate in the QA process if no information is available to AquaLog to disambiguate the query directly. Let's consider the query in Fig. 2. On the left screen we are looking for the homepage of Peter. By using string metrics the system is unable to disambiguate between Peter-Scott, Peter-Sharpe, Peter-Whalley, etc. Therefore, user feedback is required. Moreover, on the right screen user feedback is required to disambiguate the term "homepage" (is the same as "has-web-address") as it is the first time the system came across this term, no synonyms have been identified in WordNet that relate both labels together, and the ontology does not provide further ways to disambiguate. We ask the system to learn the user's vocabulary and context for future occasions.

In Fig. 3 we are asking for the web address of Peter, who has an interest in semantic web. In this case AquaLog does not need any assistance from the user, given that, by analyzing the ontology, only one of the "Peters" has an interest in Semantic Web, and only one possible ontology relation, "has-research-interest" (taking into account taxonomy inheritance) exists between "person" and the concept "research area", of which "semantic web" is an instance. Also the similarity relation between "homepage" and "has-web-address" has been learned by the Learning Mechanism in the context of that query arguments, so the performance of the system improves over the time. When the RSS comes across a similar query it has to access the ontology information to recreate the context and complete the ontology triples. In that way, it realizes that the second part of the query "who has an interest on the Semantic Web" is a modifier of the term "Peter".

3. AquaLog architecture

AquaLog is implemented in Java as a modular web application, using a client–server architecture. Moreover, AquaLog provides an API, which allows future integration in other platforms and independent use of its components. Fig. 4 shows

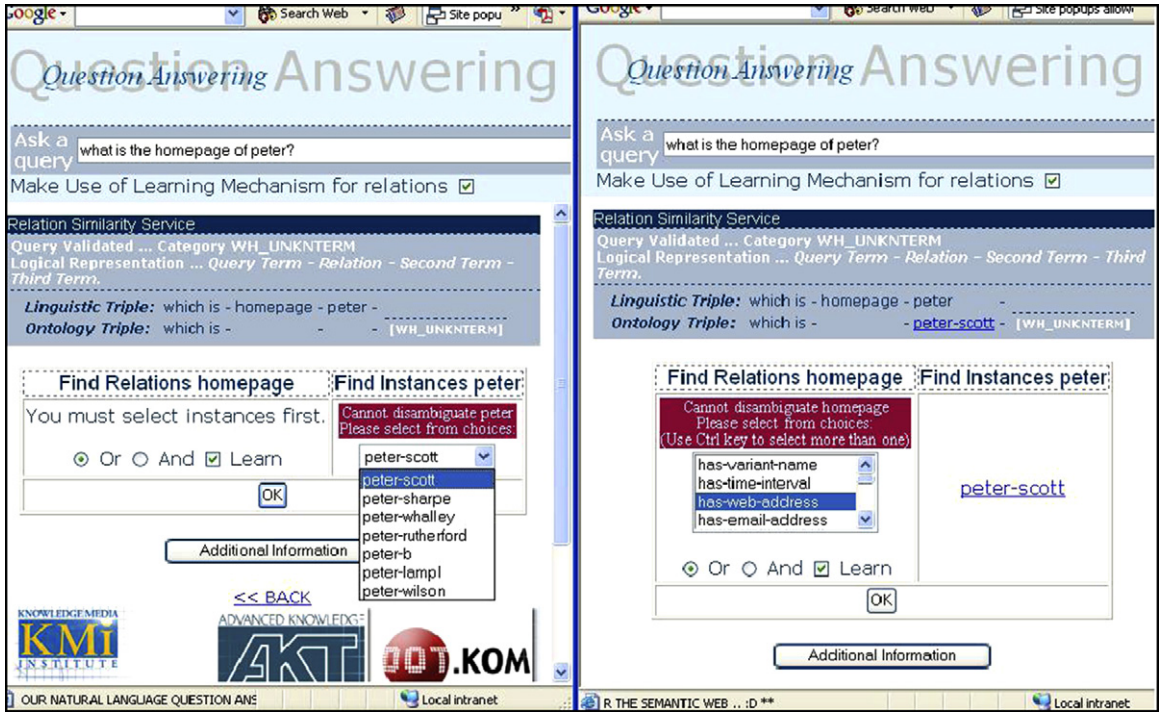

Fig. 2. Illustrative example of user interactivity to disambiguate a basic query.

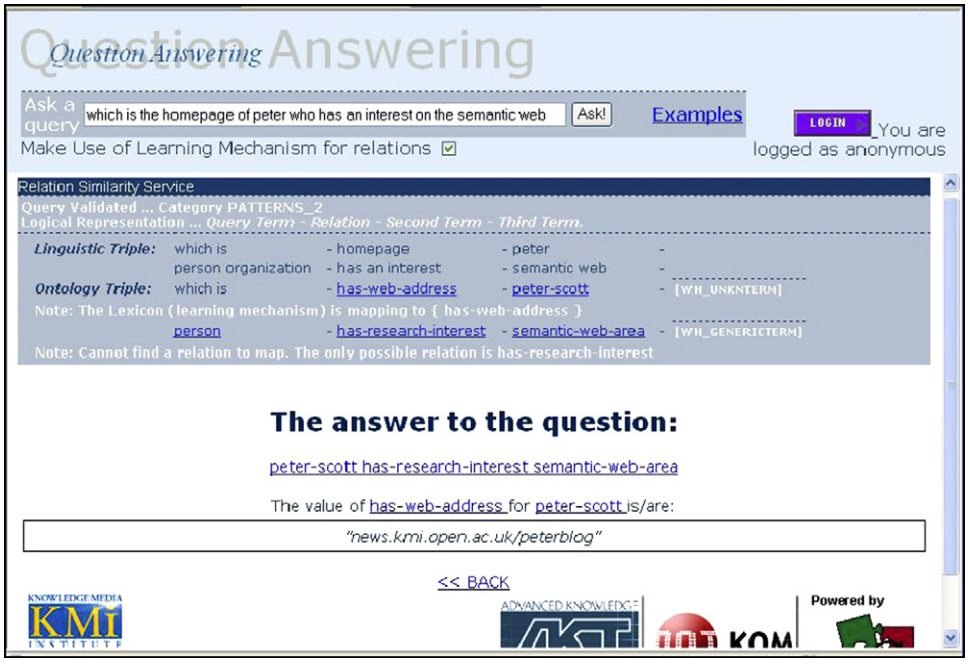

Fig. 3. Illustrative example of AquaLog disambiguation.

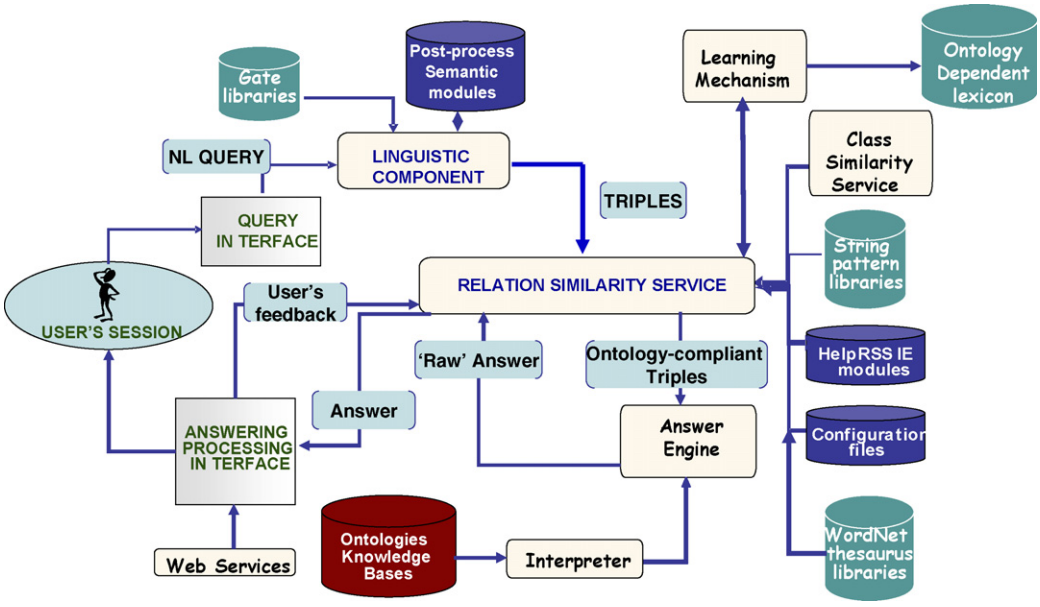

Fig. 4. The AquaLog global architecture.

the different components of the AquaLog architectural solution. As a result of this design, our existing version of AquaLog is modular, flexible and scalable.[1]

The Linguistic Component and the Relation Similarity Service (RSS) are the two central components of AquaLog, both of them portable and independent in nature. Currently AquaLog automatically classifies the query following a linguistic criterion. The linguistic coverage can be extended through the use of regular expressions by augmenting the patterns covered by an existing classification or creating a new one. This functionality is discussed in Section 4.

A key feature of AquaLog is the use of a plug-in mechanism, which allows AquaLog to be configured for different Knowledge Representation (KR) languages. Currently we subscribe to the Operational Conceptual Modeling Language (OCML), using our own OCML-based KR infrastructure [52]. However, in future our aim is also to provide direct plug-in mechanisms for the emerging RDF and OWL servers.[2]

[1] AquaLog is available for developers as Open Source licensed under the Apache License, Version 2.0 https://sourceforge.net/projects/aqualog.

[2] We are able to import and export RDF(S) and OWL from OCML, so the lack of an explicit RDF/OWL plug-in is not a problem in practice.

To reduce the number of calls/requests to the target knowledge base and to guarantee real-time question answering, even when multiple users access the server simultaneously, the AquaLog server accesses and caches basic indexing data from the target Knowledge Bases (KBs) at initialization time. The cached information in the server can be efficiently accessed by the remote clients. In our research department, KMi, since AquaLog is also integrated into the KMi Semantic Portal [35], the KB is dynamically and constantly changing with new information obtained from KMi websites through different agents. Therefore, a mechanism is provided to update the cached indexing data on the AquaLog server. This mechanism is called by these agents when they update the KB.

As regards portability, the only entry points which require human intervention for adapting AquaLog to a new domain are the configuration files. Through the configuration files it is possible to specify the parameters needed to initialize the AquaLog server. The most important parameters are: the ontology name and server, login details if necessary, the name of the plug-in, and slots that correspond to pretty names. Pretty names are alternative names that an instance may have. Optionally, the main concepts of interest in an ontology can be specified.

The other ontology-dependent parameters required in the configuration files are the ones which specify that the term "who", "where", "when" corresponds, for example, to the ontology terms "person/organization", "location" and "time-position", respectively. The independence of the application from the ontology is guaranteed through these parameters.

There is no single strategy by which a query can be interpreted, because this process is highly dependent on the type of the query. So, in the next sections we will not attempt to give a comprehensive overview of all the possible strategies, but rather we will show a few examples, which are illustrative of the way the different AquaLog components work together.

3.1. AquaLog triple approach

Core to the overall architecture is the triple-based data representation approach. A major challenge in the development of the current version of AquaLog is to efficiently deal with complex queries in which there could be more than one or two terms. These terms may take the form of modifiers that change the meaning of other syntactic constituents, and they can be mapped to instances, classes, values, or combinations of them, in compliance with the ontology to which they subscribe. Moreover, the wider expressiveness adds another layer of complexity mainly due to the ambiguous nature of human language. So, naturally the question arises of how far the engineering functionality of the triple-based model can be extended to map a triple obtained through a NL query into a triple that can be realized by a given ontology.

To accomplish this task, the triple in the current AquaLog version has been slightly extended. Therefore an existing linguistic triple now consists of one, two or even three terms connected through a relationship. A query can be translated into one or more linguistic-triples, and then each linguistic triple can be translated into one or more ontology-compliant-triples. Each triple also has additional lexical features in order to facilitate reasoning about the answer, such as the voice and tense of the relation. Another key feature for each triple is its *category* (see an example table of query types in Appendix A in Section 11). These *categories* identify different basic structures of the NL query and their equivalent representation in the triple. Depending on the *category*, the triple tells us how to deal with its elements, what inference process is required and what kind of answer can be expected. For instance, different queries may be represented by triples of the same category, since, in natural language, there can be different ways of asking the same question, i.e. "who works in akt?"[3] and "Show me all researchers involved in the akt project". The classification of the triple may be modified during its life cycle in compliance with the target ontology it subscribes to.

In what follows we provide an overview of each component of the AquaLog architecture.

4. Linguistic component: from questions to query triples

When a query is asked, the Linguistic Component's task is to translate from NL to the triple format used to query the ontology (Query-Triples). This preprocessing step helps towards the accurate classification of the query and its components by using standard research tools and query classification. Classification identifies the type of question and hence the kind of answer required. In particular, AquaLog uses the GATE [15,49] infrastructure and resources (language resources, processing resources like ANNIE, serial controllers, pipelines, etc.) as part of the Linguistic Component. Communication between AquaLog and GATE takes place through the standard GATE API.

After the execution of the GATE controller, a set of syntactic annotations associated with the input query are returned. In particular, AquaLog makes use of GATE processing resources for *English tokenizer*, *sentence splitter*, *POS tagger* and *VP chunker*. The annotations returned after the sequential execution of these resources include information about sentences, tokens, nouns and verbs. For example, we get voice and tense for the verbs and categories for the nouns, such as determinant, singular/plural, conjunction, possessive, determiner, preposition, existential, wh-determiner, etc. These features, returned for each annotation, are important to create the triples, for instance, in the case of verb annotations, together with the voice and sense it also includes information that indicates if it is the main verb of the query (the one that separates the nominal group and the predicate). This information is used by the Linguistic Component to establish the proper attachment of prepositional phrases previous to the creation of the Query-Triples (an example is shown in Section 5.3).

When developing AquaLog we extended the set of annotations returned by GATE, by identifying noun terms, relations,[4] question indicators (which/who/when, etc.) and patterns or types

[3] AKT is a EPSRC founded project in which the Open University is one of the partners. http://www.aktors.org/akt/.

[4] Hence, for example we exploited the fact that natural language commonly uses a preposition to express a relationship. For instance, in the query "who has

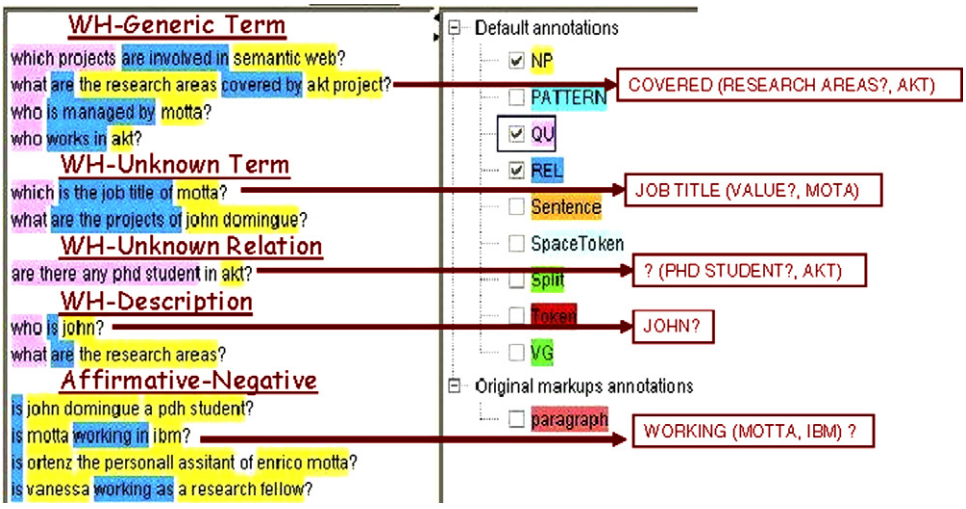

Fig. 5. Example of GATE annotations and linguistic triples for basic queries.

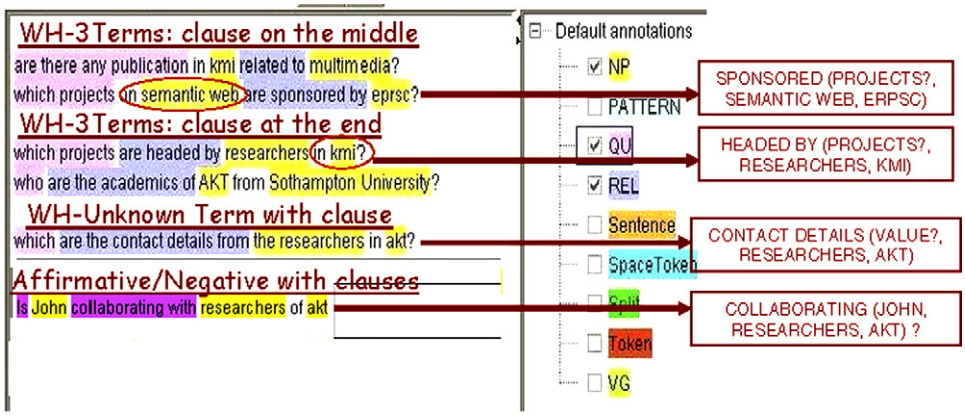

Fig. 6. Example of GATE annotations and linguistic triples for basic queries with clauses.

of questions. This is achieved by running, through the use of GATE *JAPE transducers*, a set of *JAPE grammars* that we wrote for AquaLog. JAPE is an expressive, regular expression based rule language offered by GATE[5] (see Section 4.2 for an example of the JAPE grammars written for AquaLog). These *JAPE grammars* consist of a set of *phases*, that run sequentially, and each phase is defined as a set of pattern rules, which allow us to recognize regular expressions using previous annotations in documents. In other words, the *JAPE grammars*' power lie in their ability to regard the data stored in the GATE annotation graphs as simple sequences, which can be matched deterministically by using regular expressions.[6] Examples of the annotation obtained after the use of our JAPE grammars can be seen in Figs. 5–7.

Currently, the linguistic component, through the *JAPE grammars*, dynamically identifies around 14 different question types or intermediate representations (examples can be seen in Appendix A of this paper), including: basic queries requiring an affirmation/negation or a description as an answer; or the big set of queries constituted by a wh-question, like "are there any phd students in dotkom?" where the relation is implicit or unknown or "which is the job title of John?" where no information about the type of the expected answer is provided, etc.

Categories not only tell us the kind of solution that needs to be achieved, but also they give an indication of the most likely common problems that the Linguistic Component and Relation Similarity Services will need to deal with to understand this particular NL query and in consequence it guides the process of creating the equivalent intermediate representation or Query-Triple. For instance, in the previous example "*what are the research areas covered by the akt project*?" thanks to the category the Linguistic Component knows that it has to create one triple that represents a explicit relationship between an explicit generic term and a second term. It gets the annotations for the *query terms*, *relations* and *nouns* preprocesses them and generates the following Query-Triple: ⟨*research areas*, *covered*, *akt project*⟩ in which "research areas" has correctly been identified as the query term (instead of "what").

For the intermediate representation, we use the triple-based data model rather than logic, mainly because at this stage we do

a research interest in semantic services in KMi?" the relation of the query is a combination of the verb "has" and the noun "research interest".

[5] http://gate.ac.uk/sale/tao/index.html.

[6] LC binaries and JAPE grammars can be downloaded http://kmi.open.ac.uk/technologies/aqualog.

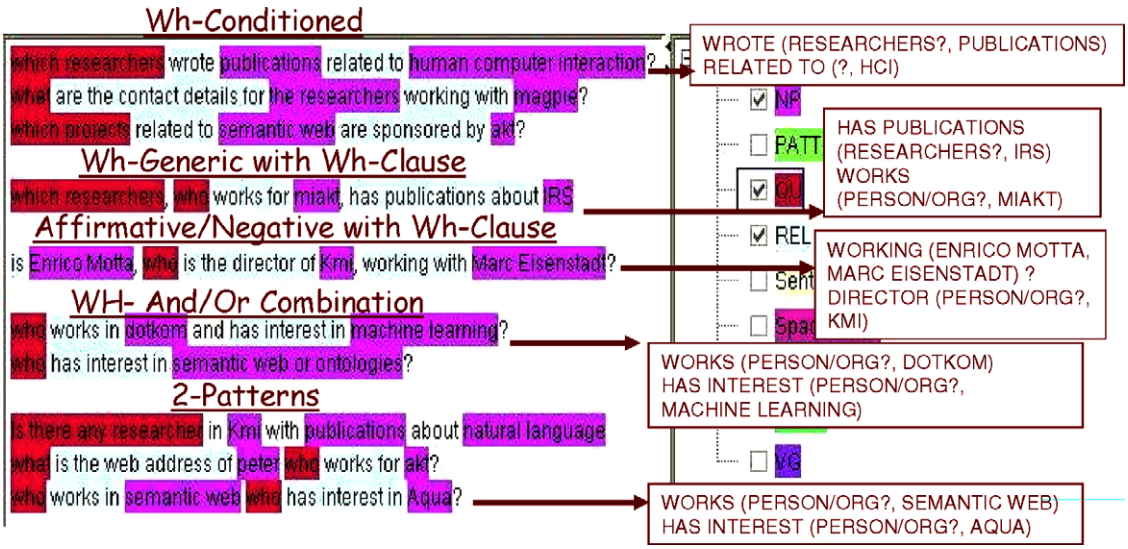

Fig. 7. Example of GATE annotations and linguistic triples for combination of queries.

not have to worry about getting the representation completely right. The role of the intermediate representation is simply to provide an easy and sufficient way to manipulate input for the RSS. An alternative would be to use more complex linguistic parsing, but as discussed by Katz et al. [33,29], although parse trees (as for example, the NLP parser for Stanford [34]) capture syntactic relations and dependencies, they are often time-consuming and difficult to manipulate.

4.1. Classification of questions

Here we present the different kind of questions and the rationale behind distinguishing these categories. We are dealing with basic queries (Section 4.1.1), basic queries with clauses (Section 4.1.2) and combinations of queries (Section 4.1.3). We considered these three main groups of queries based on the number of triples needed to generate an equivalent representation of the query.

It is important to emphasize that, at this stage, all the terms are still strings or arrays of strings, without any correspondence with the ontology. This is because the analysis is completely domain independent and is entirely based on the GATE analysis of the English language. The Query-Triple is only a formal, simplified way of representing the NL-query. Each triple represents an explicit relationship between terms.

4.1.1. Linguistic procedure for basic queries

There are several different types of basic queries: we can mention the affirmative negative query type, which are those queries that requires an affirmation or negation as an answer, i.e. "is John Domingue a phd student?", "is Motta working in ibm?" Another big set of queries are those constituted by a "wh-question", such as the ones starting with: what, who, when, where, are there any, does anybody/anyone or how many. Also imperative commands like list, give, tell, name, etc.; are treated as wh-queries (see an example table of query types in Appendix A in Section 11).

In fact, wh-queries are categorized depending on the equivalent intermediate representation. For example, in "who are the academics involved in the semantic web?" the generic triple will be of the form ⟨generic term, relation, second term⟩, in our example ⟨academics, involved, semantic web⟩. A query with an equivalent triple representation is "which technologies has KMi produced?", where the triple will be technologies, has produced, KMi⟩. However, a query like "are there any Phd students in akt?" has another equivalent representation, where the relation is implicit or unknown ⟨phd students, ?, akt⟩. Other queries may provide little or no information about the type of the expected answer, e.g. "what is the job title of John?", or they can be just a generic enquiry about someone or something, e.g. "who is Vanessa?", "what is an ontology?" (see Fig. 5).

4.1.2. Linguistic procedure for basic queries with clauses (three-terms queries)

Let us consider the request "List all the projects in the knowledge media institute about the semantic web", where both "in knowledge media institute" and "about semantic web" are modifiers (i.e. they modify the meaning of other syntactic constituents). The problem here is to identify the constituent to which each modifier has to be attached. The Relation Similarity Service (RSS) is responsible for resolving this ambiguity through the use of the ontology, or in an interactive way by asking for user feedback. The linguistic component's task is therefore to pass the ambiguity problem to the RSS through the intermediate representation, as part of the translation process.

We must remember that all these queries are always represented by just one Query-Triple at the linguistic stage. In the case that modifiers are presented, the triple will consist of three terms instead of two, for instance, in "are there any planet news written by researchers from akt?", in which we get the terms "planet news", "researchers" and "akt" (see Fig. 6).

The resultant Query-Triple is the input for the Relation Similarity Services that process the basic queries with clauses as explained in Section 5.2.

4.1.3. Linguistic procedure for combination of queries

Nevertheless, a query can be a composition of two explicit relationships between terms, or a query can be a composition of two basic queries. In this case, the intermediate representation usually consists of two triples, one triple per relationship. For instance, in "are there any planet news written by researchers working in akt?", where the two linguistic triples will be ⟨planet news, written, researchers⟩ and ⟨*which are*, working, akt⟩.

Not only are these complex queries classified depending on the kind of triples they generate but also the resultant triple *categories* are the driving force to generate an answer by combining the triples in an appropriate way.

There are three ways in which queries can be combined. Firstly, queries can be combined by using a "and" or "or" conjunction operator, as in "which projects are funded by epsrc and are about semantic web?" This query will generate two Query-Triples: ⟨funded (projects, epsrc)⟩ and ⟨about (projects, semantic web)⟩ and the subsequent answer will be a combination of both lists obtained after resolving each triple.

Secondly, a query may be conditioned to a second query, as in "which researchers wrote publications related to social aspects?" where the second clause modifies one of the previous terms. In this example, and at this stage, ambiguity cannot be solved by linguistic procedures; therefore the term to be modified by the second clause remains uncertain.

Finally, we can have combinations of two basic patterns, e.g. "what is the web address of Peter who works for akt?" or "which are the projects led by Enrico which are related to multimedia technologies?" (see Fig. 7).

Once the intermediate representation is created, prepositions and auxiliary words, such as: "in", "about", "of", "at", "for", "by", "the", "does", "do", "did" are removed from the Query-Triples because in the current AquaLog implementation their semantics are not exploited to provide any additional information to help in the mapping between Query-Triples into the Ontology-compliant triples (Onto-Triples).

The resultant Query-Triple is the input for the Relation Similarity Services that process the combinations of queries as explained in Section 5.3.

4.2. The use of JAPE grammars in AquaLog: illustrative example

Consider the basic query in Fig. 5 "what are the research areas covered by the akt project?" as an illustrative example in the use of JAPE grammars to classify the query and obtain the correct annotations to create a valid triple. Two *JAPE grammar* files were written for AquaLog. The first one in the GATE execution list annotates the nouns: "researchers" and "akt projects"; relations: "are" and "covered by"; and query terms: "what" (see Fig. 8). For instance, consider the rule *Rule:NP* in Fig. 8. The pattern on the left hand side (i.e., before "→") identifies noun phrases. Noun phrases are word sequences that start with zero or more determiners (identified by the (DET)* part of the pattern). Determiners can be followed by zero or more adverbs and adjectives, nouns or coordinating conjunctions in any order (identified by the ((RB)*ADJ)|NOUN|CC)* part of the pattern). A noun phrase mandatorily finishes with a noun (NOUN). RB, ADJ, NOUN and CC are macros and act as placeholders for other rules. For example, the macro *LIST* used by the rule *Rule:QU1* or the macro *VERB* used by rule *Rule:REL1* in Fig. 8. The macro *VERB* contains the disjunction of seven patterns, which means that the macro will fire if a word satisfies any of these seven patterns, e.g. last pattern identified words assigned to the *VB*

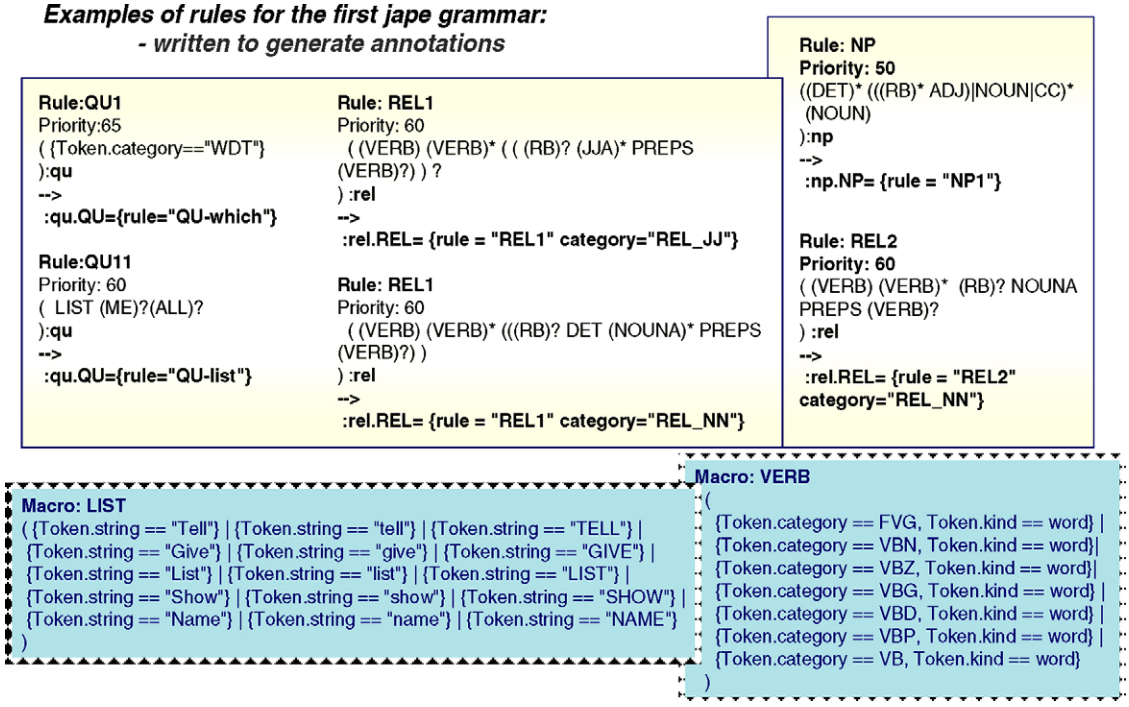

Fig. 8. Set of JAPE rules used in the query example to generate annotations.

Examples of rules for the second jape grammar:
- written to classify the query using previous annotations

```
Rule:PATTERN_whgenericterm
Priority:65
(  (({Token.category==IN})? QUWhichClass (AUX|AUX_EXT) TERM RELATION) |
(QUWhichClass (Modal)? TERM RELATION) | (QUWhichClass NOUN (NOUN)* RELATION) |
  (QUWhich IS_ARE TERM (QUWhich)? RELATION TERM) |
  (QUWho IS_ARE TERM RELATION) |
  (QUWhichClass (QUWhich|QU-who-what|Modal)? RELATION TERM) |
  (QU-who-what AUX TERM (QUWhich|QU-who-what)? RELATION TERM) |
  (QUWho RELATION TERM) |
  (QU-anybody (QUWhich|QU-who-what)? RELATION TERM) |
  (QUWhe AUX TERM RELATION) |
  (QUWhe RELATION TERM) |
  (QUListClass (QUWhich|QU-who-what)? RELATION TERM)
):pattern
-->
 :pattern.PATTERN={rule="PATTERN_whgenericterm" category="wh-genericterm"}
```

```
Macro: QUWhich
(
{QU.rule == "QU-which"} |
{QU.rule == "QU-list"}
)
```

```
Macro: RELATION
(
{REL.rule == “REL1"} |
{REL.rule == “REL2"}
)
```

```
Macro: TERM
(
{NP.rule == “NP1"}
)
```

Fig. 9. Set of JAPE rules used in the query example to classify the queries.

POS tags. POS tags are assigned in the “*category*” feature of a “*Token*” annotation used in GATE to encode the information about the analyzed question. Any word sequence identified by the left hand side of a rule can be referenced in its right hand size, e.g. *rel.REL* = {*rule* = “*REL1*”...} is referenced by the macro *RELATION* in the second grammar (Fig. 9).

The second grammar file based on the previous annotations (rules) classifies the example query as “*wh-genericterm*”. The set of *JAPE rules* used to classify the query can be seen in Fig. 9. The pattern fired for that query “*QUWhich IS_ARE TERM RELATION TERM*” is marked in bold. This pattern relies on the macros *QUWhich*, *IS_ARE*, *TERM and RELATION*.

Thanks to this architecture, that takes advantage of the *JAPE grammars*, although we can still only deal with a subset of natural language, it is possible to extend this subset in a relatively easy way by updating the regular expressions in the *JAPE grammars*. This design choice ensures the easy portability of the system with respect to both ontologies and natural languages.

5. Relation similarity service: from triples to answers

The RSS is the backbone of the question-answering system. The RSS component is invoked after the NL query has been transformed into a term-relation form and classified into the appropriate category. The RSS is the main component responsible for generating an ontology-compliant logical query. Essentially, the RSS tries to make sense of the input query by looking at the structure of the ontology and the information stored in the target KBs, as well as using string similarity matching, generic lexical resources such as WordNet, and a domain-dependent lexicon obtained through the use of a Learning Mechanism, explained in Section 5.5.

An important aspect of the RSS is that it is interactive. In other words, when ambiguity arises between two or more possible terms or relations the user will be required to interpret the query.

Relation and concept names are identified and mapped within the ontology through the RSS and the Class Similarity Service (CSS) respectively (the CSS is a component which is part of the Relation Similarity Service). Syntactic techniques like string algorithms are not useful when the equivalent ontology terms have dissimilar labels from the user term. Moreover, relations names can be considered as “second class citizens”, the rationality behind this is that mapping relations based on its name (labels) is more difficult that mapping concepts. Labels, in the case of concepts, often catch the meaning of the semantic entity, while in relations its meaning is given by the type of its domain and its range rather than by its name (typically vaguely defined as, e.g. “related to”), with the exception of the cases in which the relations are presented in some ontologies as a concept (e.g. *has-author* modeled as a concept *Author* in a given ontology). Therefore the similarity services should use the ontology semantics to deal with these situations.

Proper names, instead, are normally mapped into instances by means of string metric algorithms. The disadvantage of such an approach is that without some facilities for dealing with unrecognized terms, some questions are not accepted. For instance, a question like “is there any researcher called Thompson?” would not be accepted if Thompson is not identified as an instance in the current KB. However, a partial solution is implemented for affirmative/negative types of questions, where we make sense of questions in which only one of two instances is recognized, e.g. in the query “is Enrico working in ibm?”,

where "Enrico" is mapped into "Enrico-motta" in the KB, but "ibm" is not found. The answer in this case, is an indirect negative answer, namely the place were Enrico Motta is working.

In any non-trivial natural language system, it is important to deal with the various sources of ambiguity and the possible ways of treating them. Some sentences are syntactically (structurally) ambiguous and although general world knowledge does not resolve this ambiguity, within a specific domain it may happen that only one of the interpretations is possible. The key issue here is to determine some constraints derived from the domain knowledge and to apply them in order to resolve ambiguity [14]. Where the ambiguity cannot be resolved by domain knowledge the only reasonable course of action is to get the user to choose between the alternative readings.

Moreover, since every item on the Onto-Triple is an entry point in the knowledge base or ontology, they are also clickable, giving the user the possibility to get more information about them. Also, AquaLog scans the answers for words denoting instances that the system "knows about", i.e. which are represented in the knowledge base, and then adds hyperlinks to these words/phrases, indicating that the user can click on them and navigate through the information. In fact, the RSS is designed to provide justifications for every step of the user interaction. Intermediate results obtained through the process of creating the Onto-Triple are stored in auxiliary classes, which can be accessed within the triple.

Note that the category to which each Onto-Triple belongs can be modified by the RSS during its life cycle, in order to satisfy the appropriate mappings of the triple within the ontology.

In what follows, we will show a few examples, which are illustrative of the way the RSS works.

5.1. RSS procedure for basic queries

Here we describe how the RSS deals with basic queries. For example, let's consider a basic question like "what are the research areas covered by akt?" (see Figs. 10 and 11). The query is classified by the Linguistic component as a basic generic-type (one *wh-query* represented by a triple formed by an explicit binary relationship between two define terms) as shown is Section 4.1.1. Then, the first step for the RSS is to identify that "research areas" is actually a "research area" in the target KB and "akt" is a "project" through the use of string distance metrics and WordNet synonyms if needed. Whenever a successful match is found, the problem becomes one of finding a relation which links "research areas" or any of its subclasses to "projects" or any of its superclasses.

By analyzing the taxonomy and relationships in the target KB, AquaLog finds that the only relations between both terms are "addresses-generic-area-of-interest" and "uses-resource". Using the WordNet lexicon, AquaLog identifies that a synonym for "covered" is "addresses", and therefore suggests to the user that the question could be interpreted in terms of the relation "addresses-generic-area-of-interest". Having done this, the answer to the query is provided. It is important to note that in order to make sense of the triple ⟨research areas, covered, akt⟩, all the subclasses of the generic term "research areas" need to be considered. To clarify this, let's consider a case in which we are talking about a generic term "person", then the possible relation could be defined only for researchers, students or academics, rather than people in general. Due to the inheritance of relations through the subsumption hierarchy a similar type of analysis is required for all the super-classes of the concept "projects".

Whenever multiple relations are possible candidates for interpreting the query, if the ontology does not provide ways to further

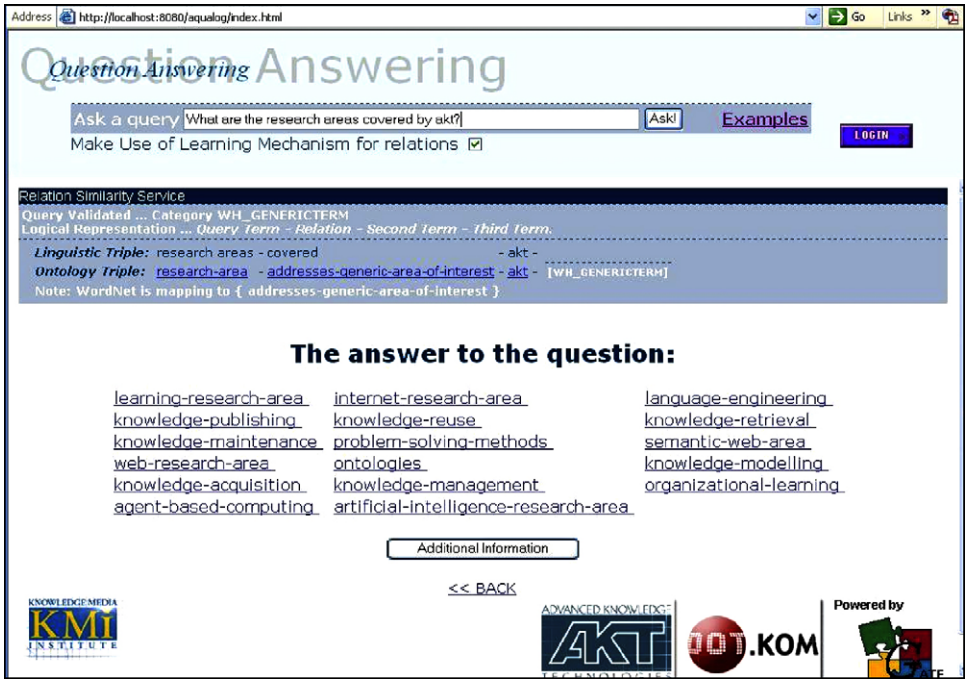

Fig. 10. Example of AquaLog in action for basic generic-type queries.

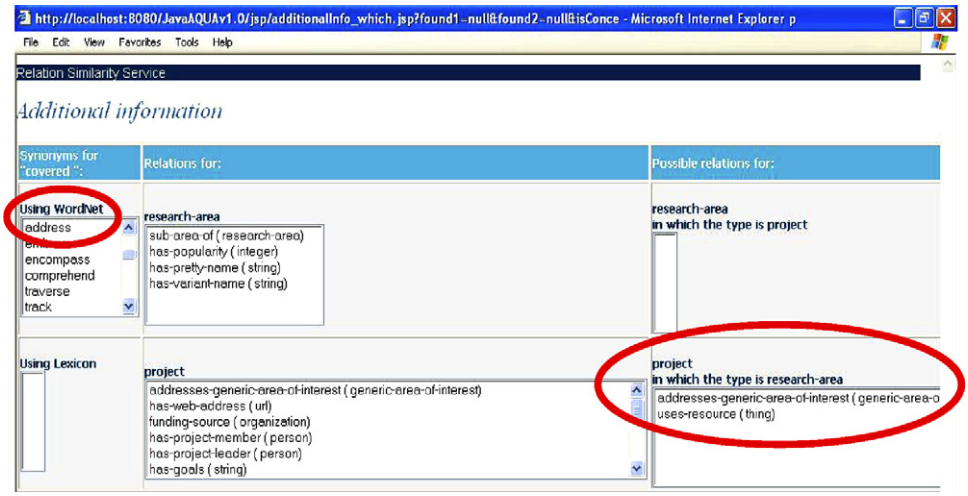

Fig. 11. Example of AquaLog additional information screen from Fig. 10 example.

discriminate between them, string matching is used to determine the most likely candidate, using the relation name, the learning mechanism, eventual aliases, or synonyms provided by lexical resources such as WordNet [21]. If no relations are found by using these methods, then the user is asked to choose from the current list of candidates.

A typical situation the RSS has to cope with is one in which the structure of the intermediate query does not match the way the information is represented in the ontology. For instance, the query "who is the secretary in Knowledge Media Inst?", as shown in Fig. 12, may be parsed into ⟨person, secretary, kmi⟩, following purely linguistic criteria, while the ontology may be organized in terms of ⟨secretary, works-in-unit, kmi⟩. In these cases the RSS is able to reason about the mismatch, re-classify the intermediate query and generate the correct logical query. The procedure is as follows. The question "who is the secretary in Knowledge Media Inst?", is classified as a "basic generic type" for the Linguistic Component. The first step is to identify that *KMi* is a "research institute", which is also part of an "organization", in the target KB, and *who* could be pointing to a "person" (sometimes an "organization"). Similarly to the previous example, the problem becomes one of finding a relation which links a "person" (or any of its subclasses like "academic", "students"...) to a "research institute" (or an "organization"), but in this particular case, there is also a successful matching for "secretary" in the KB, in which "secretary" is not a relation but a subclass of "person", and therefore the triple is classified from a generic one formed by a binary relation "secretary" between the terms "person" and "research institute" to a generic one in which the relation is unknown or implicit between the terms "secretary", which is more specific than "person" and "research institute".

If we take the example question presented in Ref. [14] "who takes the database course?" it may be that we are asking for "lecturers who teach databases" or for "students who study databases". In this cases, AquaLog will ask the user to select the appropriate relation within all the possible relations between a generic person (lecturers, students, secretaries...) and a course. Therefore, the translation process from lexical to ontological triples can go ahead without misinterpretations.

Fig. 12. Example of AquaLog in action for relations formed by a concept.

Note that some additional lexical features, which help in the translation or put a restriction on the answer, presented in the Onto-Triple are: if the relation is passive, or reflexive or the relation is formed by "is". The last means that the relation cannot be an "ad hoc" relation between terms but is a relation of type "subclass-of" between terms related in the taxonomy, as in the query "is John Domingue a PhD student?"

Also, a particular lexical feature is whether the relation is formed by verbs or by a combination of verbs and nouns; this feature is important because in the case that the relation contains nouns the RSS has to make additional checks to map the relation in the ontology. For example, consider the relation "is the secretary", the noun "secretary" may correspond to a concept in the target ontology, while, e.g. the relation "is the job title" may correspond to the slot "has-job-title" for the concept person.

5.2. RSS procedure for basic queries with clauses (three-term queries)

When we described the Linguistic Component in Section 4.1.2, we discussed that in the case of the basic queries in which there is a modifier clause, the triple generated is not in the form of a binary-relation but a ternary-relation. That is to say that from one Query-Triple composed of three terms, two Onto-Triples are generated by the RSS. In these cases the ambiguity can also be related to the way the triples are linked.

Depending on the position of the modifier clause (consisting of a preposition plus a term) and the relation within the sentence, we can have two different classifications, one in which the clause is related to the first term by an implicit relation, for instance "are there any projects *in KMi* related to natural language?" or "are there any projects *from researchers* related to natural language?"; and one where the clause is at the end and after the relation, as in "which projects are headed by researchers *in akt*?" or "what are the contact details for researchers *in akt*?" The different classifications or query types determine whether the verb is going to be part of the first triple as in the latter example or part of the second triple as in the former example. However, at a linguistic level it is not possible to disambiguate the term in the sentence which the last part of the sentence, "related to natural language" (and "in akt" in the previous examples), refers to.

The RSS analysis relies on its knowledge of the particular application domain to solve such modifier attachment. Alternatively, the RSS may attempt to resolve the modifier attachment by using heuristics.

For example, to handle the previous query "are there any projects on the semantic web sponsored by epsrc", the RSS uses a heuristic which suggests attaching the last part of the sentence "sponsored by epsrc" to the closest term that is represented by a class or non-ground term in the ontology (e.g. in this case the class "project"). Hence the RSS will create the following two Onto-Triples: a generic one ⟨project, sponsored, epsrc⟩ and one in which the relation is implicit ⟨project, ?, semantic web⟩. The answer is a combination of a list of projects related to semantic web and a list of proj ects sponsored by epsrc (Fig. 13).

However, if we consider the query "which planet news stories are written by researchers in akt?" and apply the same heuristic to disambiguate the modifier "in akt", the RSS will select the non-ground term "researchers" as the correct attachment for "akt". Therefore, to answer this query, first it is necessary to get the list of researchers related to akt, and then to get a list of planet news stories for each of the researchers. It is important to notice that a relation in the linguistic triple may be mapped into more than one relation in the Onto-Triple, as for example "are written" is translated into "owned-by" or "has-author".

The use of this heuristic to attach the modifier to the non-ground term can be appreciated comparing the ontology triples created in Fig. 13 ("Are there any project on semantic web sponsored by eprsc?") and Fig. 14 ("Are there any project by researcher working in AKT?"). For the former, the modifier "sponsored by eprsc", is attached to the query term in the first triple, i.e. "project". For the later, the modifier "working in akt" is attached to the second term in the first triple, i.e., "researchers".

5.3. RSS procedure for combination of queries

Complex queries generate more than one intermediate representation as shown in the linguistic analysis in Section 4.1.3.

Fig. 13. Example of AquaLog in action for queries with a modifier clause in the middle.

Fig. 14. Example of AquaLog in action for queries with a modifier clause at the end.

Depending on the type of queries the triples are combined following different criteria.

To clarify the kind of ambiguity we deal with, let's consider the query "who are the researchers in akt, who have interest in knowledge reuse". By checking the ontology, or, if needed, by using user feedback, the RSS module can map the terms in the query either to instances or to classes in the ontology, in a way that it can determine that "akt" is a "project" and therefore, it is not an instance of the classes "persons" or "organizations" or any of their subclasses. Moreover, in this case, the relation "researchers"is mapped into a "concept" in the ontology which is a subclass of "person", and therefore, the second clause "who have an interest in knowledge reuse" is without any doubt assumed to be related to "researchers". The solution in this case will be a combination of a list of researchers who work for akt and have interest in knowledge reuse (see Fig. 15).

However, there could be other situations where the disambiguation cannot be resolved by use of linguistic knowledge and/or heuristics and/or the context or semantics in the ontology, e.g. in the query "which academic works with Peter who has interest in semantic web?" In this case, since "academic" and "peter" are respectively a subclass and an instance of "per-

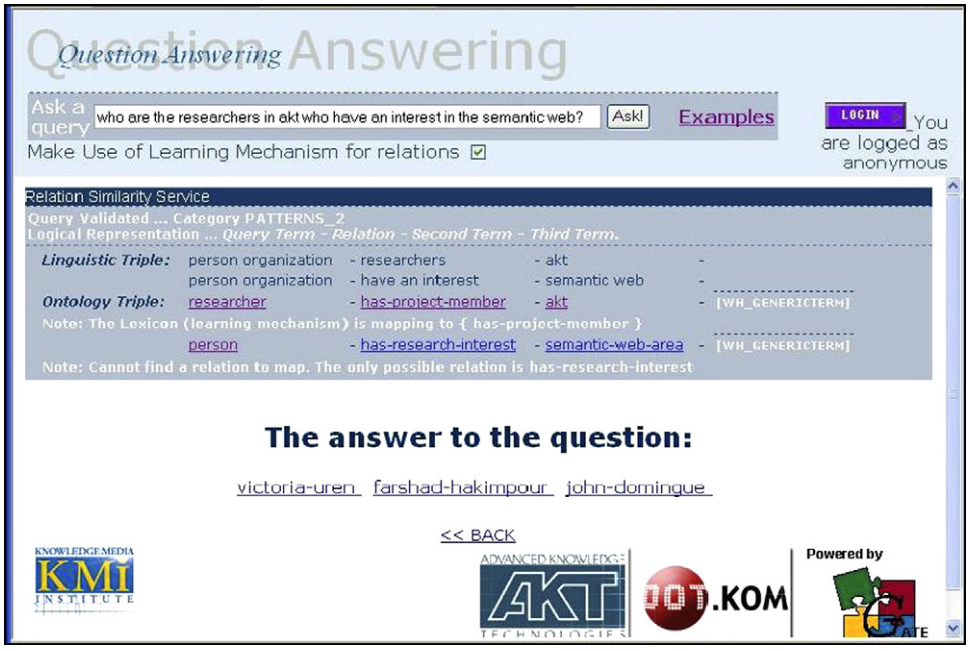

Fig. 15. Example of context disambiguation.

son", the sentence is truly ambiguous, even for a human reader.[7] In fact, it can be understood as a combination of the resulting lists of the two questions "which academic works with peter" and "which academic has an interest in semantic web", or as the relative query "which academic works with peter where the peter we are looking for has an interest in the semantic web". In such cases, user feedback is always required.

To interpret a query, different features play different roles. For example, in a query like "which researchers wrote publications related to social aspects?" the second term "publication" is mapped into a concept in our KB, therefore the adopted heuristic is that the concept has to be instantiated using the second part of the sentence "related to social aspects". In conclusion, the answer will be a combination of the generated triples: ⟨researchers, wrote, publications⟩ ⟨publications, related, akt⟩. However, frequently this heuristic is not enough to disambiguate and other features have to be used.

It is also important to underline the role that a triple element's features can play. For instance, an important feature of a relation is whether it contains the main verb of the sentence, namely the lexical verb. We can see this if we consider the following two queries: (1) "which academics work in the akt project sponsored by epsrc?"; (2) "which academics working in the akt project are sponsored by epsrc?" In both examples, the second term "akt project" is an instance, so the problem becomes to identify if the second clause "sponsored by epsrc" is related to "akt project" or to "academics". In the first example the main verb is "work", which separates the nominal group "which academics" and the predicate "akt project sponsored by epsrc", thus "sponsored by epsrc" is related to "akt project". In the second, the main verb is "are sponsored", whose nominal group is "which academics working in akt" and whose predicate is "sponsored by epsrc". Consequently, "sponsored by epsrc" is related to the subject of the previous clause, which is "academics".

5.4. Class similarity service

String algorithms are used to find mappings between ontology term names and pretty names[8] and any of the terms inside the linguistic triples (or its WordNet synonyms) obtained from the user's query. They are based on String Distance Metrics for Name-Matching Tasks, using an open-source API from Carnegie Mellon University [13]. This comprises a number of string distance metrics proposed by different communities, including edit-distance metrics, fast heuristic string comparators, token-based distance metrics, and hybrid methods. AquaLog combines the use of these metrics (it mainly uses the Jaro-based metrics) with thresholds. Thresholds are set up depending on the task of looking for concepts names, relations or instances. See Ref. [13] for a experimental comparison between the different metrics.

However, the use of string metrics and open-domain lexico-semantic resources [45], such as WordNet, to map the generic term of the linguistic triple into a term in the ontology may not be enough. String algorithms fail when the terms to compare have dissimilar labels (i.e. "researcher" and "academic"), and generic thesaurus like WordNet may not include all the synonyms and is limited in the use of nominal compounds (i.e. WordNet contains the term "municipality" but not an ontology term like "municipal-unit"). Therefore, an additional combination of methods may be used in order to obtain the possible candidates in the ontology. For instance, a lexicon can be generated manually or can be built through a learning mechanism (a similar simplified approach to the learning mechanism for relations explained in Section 6). The only requirement to obtain ontology classes that can potentially map an unmapped linguistic term is the availability of the ontology mapping for the other term of the triple. In this way, through the ontology relationships that are valid for the ontology mapped term, we can identify a set of possible candidate classes that can complete the triple. User feedback is required to indicate if one of the candidate terms is the one we are looking for, so that AquaLog, through the use of the learning mechanism, will be able to learn it for future occasions.

The WordNet component is using the Java implementation (JWNL) of WordNet 1.7.1 [29]. The next implementation of AquaLog will be coupled with the new version on WordNet, so it can make use of Hyponyms and Hypernyms. Currently, the component of AquaLog which deals with WordNet does not perform any *sense* disambiguation.

5.5. Answer engine

The Answer Engine is a component of the RSS. It is invoked when the Onto-Triple is completed. It contains the methods which take as an input the Onto-Triple, and infer the required answer to the user's queries. In order to provide a coherent answer, the category of each triple tells the answer engine not only about how the triple must be resolved (or what answer to expect) but also how triples can be linked with each other. For instance, AquaLog provides three mechanisms (depending on the triple categories) for operationally integrating the triple's information to generate an answer. These mechanisms are:

(1) And/or linking: e.g., in the query "who has an interest in ontologies or in knowledge reuse?", the result will be a fusion of the instances of people who have an interest in ontologies and the people who are interested in semantic web;
(2) Conditional link to a term: for example in "which KMi academics work in the akt project sponsored by eprsc?" the second triple ⟨*akt project*, *sponsored*, *eprsc*⟩ must be resolved and the instance representing the "akt project sponsored by eprsc" identified to get the list of academics

[7] Note that there will not be ambiguity if the user reformulates the question as "which academic works with Peter and has an interest in semantic web?", where "has an interest in semantic web" would be correctly attached to "academic" by AquaLog.

[8] Naming conventions vary depending on the KR used to represent the KB and may even change with ontologies—e.g., an ontology can have slots such as "variant name" or "pretty name". AquaLog deals with differences between KRs by means of the plug-in mechanism. Differences between ontologies need to be handled by specifying this information in a configuration file.

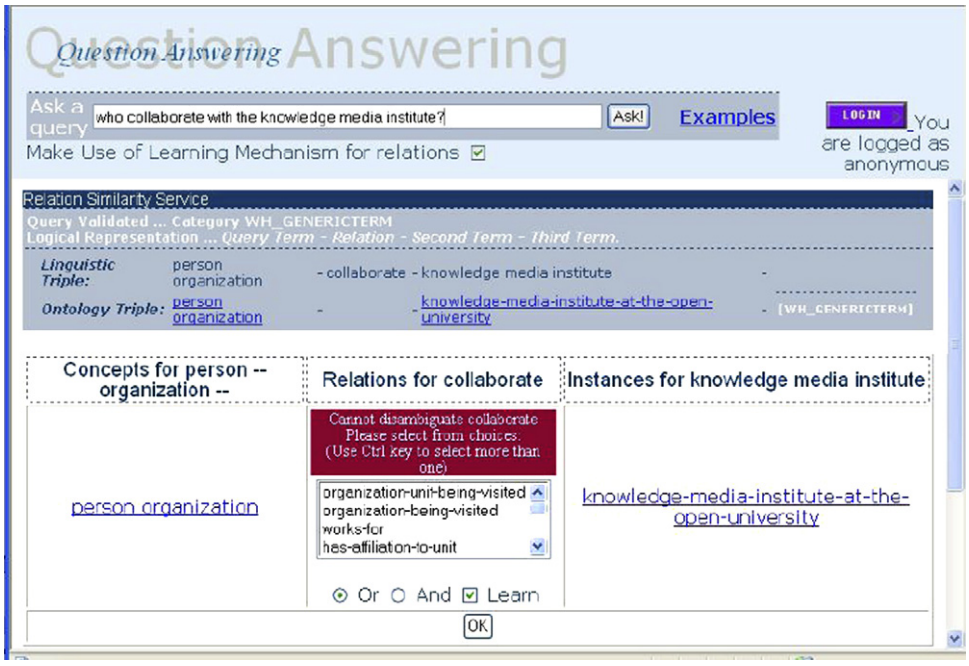

Fig. 16. Example of jargon mapping through user-interaction.

required for the first triple ⟨*KMi academics*, *work*, *akt project*⟩;

(3) Conditional link to a triple: for instance in "What is the homepage of the researchers working on the semantic web?" the second triple ⟨*researchers*, *working*, *semantic web*⟩ must be resolved and the list of researchers obtained prior to generating an answer for the first triple ⟨*?*, *homepage*, *researchers*⟩.

The solution is converted into English using simple template techniques before presenting them to the user. Future AquaLog versions will be integrated with a natural language generation tool.

6. Learning mechanism for relations

Since the universe of discourse we are working within is determined by and limited to the particular ontology used, normally there will be a number of discrepancies between the natural language questions prompted by the user and the set of terms recognized in the ontology. External resources like WordNet generally help in making sense of unknown terms, giving a set of synonyms and semantically related words which could be detected in the knowledge base. However, in quite a few cases, the RSS fails in the production of a genuine Onto-Triple because of user-specific jargon found in the linguistic triple. In such a case, it becomes necessary to learn the new terms employed by the user and disambiguate them in order to produce an adequate mapping of the classes of the ontology. A quite common and highly generic example, in our departmental ontology, is the relation *works-for*, which users normally relate with a number of different expressions: *is working*, *works*, *collaborates*, *is involved* (see Fig. 16). In all these cases the user is asked to disambiguate the relation (choosing from the set of ontology relations consistent with the question's arguments) and decide whether to learn a new mapping between his/her natural-language-universe and the reduced ontology-language-universe.

6.1. Architecture

The learning mechanism (LM) in AquaLog consists of two different methods, the *learning* and the *matching* (see Fig. 17). The latter is called whenever the RSS cannot relate a linguistic triple to the ontology or the knowledge base, while the former is always called after the user manually disambiguates an unrecognized term (and this substitution gives a positive result).

When a new item is learned, it is recorded in a database together with the relation it refers to and a series of constraints that *will determine its reuse within similar contexts*. As will be explained below, the notion of context is crucial in order to deliver a feasible matching of the recorded words. In the current version the context is defined by the arguments of the question, the name of the ontology and the user information. This set of characteristics constitutes a representation of the context and defines a structured space of hypothesis analogue[9] to that of a version space.

A version space is an inductive learning technique proposed by Mitchell [42] in order to represent the subset of all hypotheses that are consistent with the observed training examples. In our case, the training examples are given by the user-refinement (disambiguation) of the query, while the hypotheses are structured in terms of the context parameters. Moreover, a version space is

[9] Although making use of a concept borrowed from machine learning studies, we want to stress the fact that the learning mechanism *is not* a machine learning technique, in the classical sense.

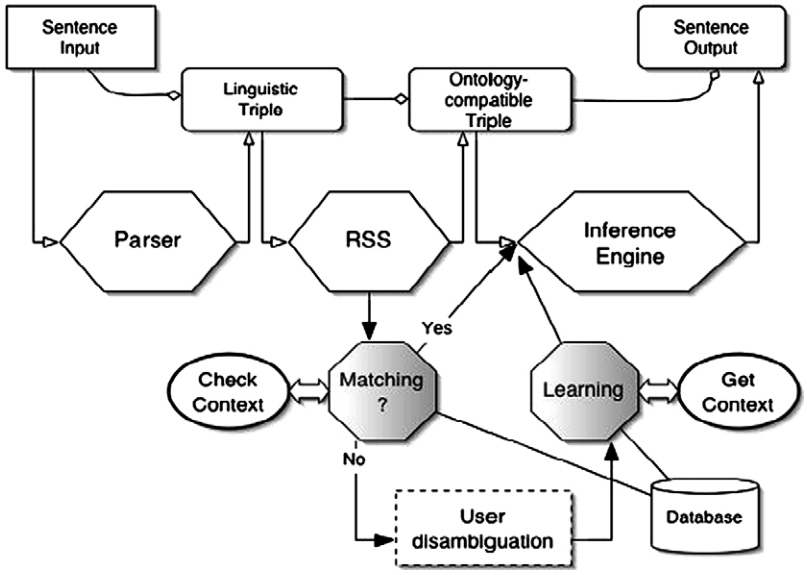

Fig. 17. The learning mechanism architecture.

normally represented just by its lower and upper bounds (maximally general and maximally specific hypothesis sets), instead of listing all consistent hypotheses. In the case of our learning mechanism these bounds are partially inferred through a generalization process (Section 6.3) that will in future be applicable also to the user-related and community derived knowledge (Section 6.4). The creation of specific algorithms for combining the lexical and the community dimensions will increase the precision of the context-definition and provide additional personalization features.

Thus, when a question with a similar context is prompted, if the RSS cannot disambiguate the relation-name, the database is scanned for some matching results. Subsequently, these results will be *context-proved* in order to check their consistency with the stored version spaces. By tightening and loosening the constraints of the version space, the learning mechanism is able to determine when to propose a substitution and when not to. For example, the user-constraint is a feature that is often bypassed, because we are inside a generic user session, or because we might want to have all the results of all the users from a single database query.

Before the matching method, we are always in a situation where the onto-triple is incomplete, the relation is unknown or it is a concept. If the new word is found in the database, the context is checked to see if it is consistent with what has been recorded previously. If this gives a positive result we can have a valid onto-triple substitution that triggers the inference engine (this latter scans the knowledge base for results); instead, if the matching fails, a user disambiguation is needed in order to complete the onto-triple. In this case, before letting the inference engine work out the results, the context is drawn from the particular question entered and it is learned together with the relation and the other information in the version space.

Of course, the matching method's movement through the ontology is opposite to the learning method's one. The latter, starting from the arguments, tries to go up until it reaches the highest valid classes possible (GetContext method), while the former takes the two arguments and checks if they are subclasses of what has been stored in the database (CheckContext method). It is also important to notice that the Learning Mechanism does not have a question classification on its own, but relies on the RSS classification.

6.2. Context definition

As said above, the notion of context is fundamental in order to deliver a feasible substitution service. In fact, two people could use the same jargon but mean different things.

For example, let's consider the question "Who collaborates with the knowledge media institute?" (Fig. 16) and assume that the RSS is not able to solve the linguistic ambiguity of the word "collaborate". The first time, some help is needed from the user, who selects "has-affiliation-to-unit" from a list of possible relations in the ontology. A mapping is therefore created between "collaborate" and "has-affiliation-to-unit", so that the next time the learning mechanism is called it will be able to recognize this specific user jargon.

Let's imagine a user, who asks the system the same question "who collaborates with the knowledge media institute?", but is referring to other research labs or academic units involved with the knowledge media institute. In fact, if asked to choose from the list of possible ontology relations, he/she would possibly enter "works-in-the-same-project".

The problem, therefore, is to maintain the two separated mappings while still providing some kind of generalization. This is achieved through the definition of the question context as determined by its coordinates in the ontology. In fact, since the referring (and *pluggable)* ontology is our universe of discourse, the context must be found within this universe. In particular, since we are dealing with triples, and in the triple what we learn is usually the relation (that is, the middle item), the context is delimited by the two arguments of the triple. In the ontology,

these correspond to two classes or two instances connected by the relation.

Therefore, in the question "who collaborates with the knowledge media institute?" the context of the mapping from "collaborates" to "has-affiliation-to-unit" is given by the two arguments "person" (in fact, in the ontology "who" is always translated into "person" or "organization") and "knowledge media institute". What is stored in the database, for future reuse, is the new word (which is also the key field in order to access the lexicon during the matching method), its mapping in the ontology, the two context-arguments, the name of the ontology and the user details.

6.3. Context generalization

This kind of recorded context is quite specific and does not let other questions benefit from the same learned mapping. For example, if afterwards we asked "who collaborates with the Edinburgh department of informatics?" we would not get an appropriate matching, even if the mapping made sense again in this case.

In order to generalize these results the strategy adopted is to record the most generic classes in the ontology, which correspond to the two triple's arguments, and, at the same time, can handle the same relation. In our case, we would store the concepts "people" and "organization-unit". This is achieved through a backtracking algorithm in the Learning Mechanism, that takes the relation, identifies its type (the type already corresponds to the highest possible class of one argument, by definition) and goes through all the connected superclasses of the other argument while checking if they can handle that same relation, with the given type. Thus, only the highest suitable classes of an ontology's branch are kept and all the questions similar to the ones we have seen will fall within the same set, since their arguments are subclasses or instances of the same concepts.

If we go back to the first example presented ("who collaborates with the knowledge media institute?"), we can see that the difference in meaning between "collaborate" → "has-affiliation-to-unit" and "collaborate" → "works-in-the-same-project" is preserved, because the two mappings entail two different contexts. Namely, in the first case, the context is given by ⟨people⟩ and ⟨organization-unit⟩, while in the second case the context will be ⟨organization⟩ and ⟨organization-unit⟩. Any other matching could not mistake the two, since what is learned is abstract but still specific enough to rule out the different cases.

A similar example can be found in the question "who are the researchers working in akt?" (see Fig. 18) the context of the mapping from "working" to "has-research-member" is given by the highest ontology classes which correspond to the two arguments "researcher" and "akt", as far as they can handle the same relation. What is stored in the database, for a future reuse, is the new word, its mapping in the ontology, the two context-arguments (in our case, we would store the concepts "people" and "project"), the name of the ontology and the user details.

Other questions which have a similar context benefit from the same learned mapping. For example, if afterwards we asked "who are the people working in buddyspace?" we would get an appropriate matching from "working" to "has-research-member".

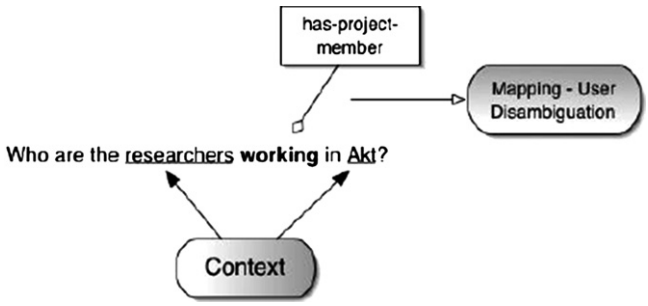

Fig. 18. Users disambiguation example.

It is also important to notice that the Learning Mechanism does not have a question classification on its own, but it relies on the RSS classification. Namely, the question is firstly recognized, and then worked out differently depending on the specific differences.

For example, we can have a generic-question learning ("who is involved in the *AKT* project?"), where the first generic argument ("Who/which") needs more processing since it can be mapped to both a person or an organization; or an unknown-relation learning ("is there any project about the semantic web?"), where the user selection and the context are mapped to the apparent lack of a relation in the linguistic triple. Also combinations of questions are taken into account, since they can be decomposed into a series of basic queries.

An interesting case is when the relation in the linguistic triple relation is not found in the ontology and is then recognized as a concept. In this case, the learning mechanism performs an iteration of the matching in order to capture the real form of the question, and queries the database as it would for an unknown relation type of question (see Fig. 19). The data that are stored in the database during the learning of a concept-case are the same as they would have been for a normal unknown-relation type of question, plus an additional field that serves as a reminder of the fact that we are within this exception. Therefore, during the matching, the database is queried as it would be for an unknown relation type, plus the constraint that is a concept. In this way, the version space is reduced only to the cases we are interested in.

6.4. User profiles and communities

Another important feature of the learning mechanism is its support for a community of users. As said above, the user details are maintained within the version space and can be considered when interpreting a query. AquaLog allows the user to enter his/her personal information and thus to log in and start a session where all the actions performed on the learned lexicon table are also strictly connected to his/her profile (see Fig. 20). For example, during a specific user-session it is possible to delete some previously recorded mappings, an action that is not permitted to the generic user. This latter has the roughest access to the learned material: having no constraints on the user field,

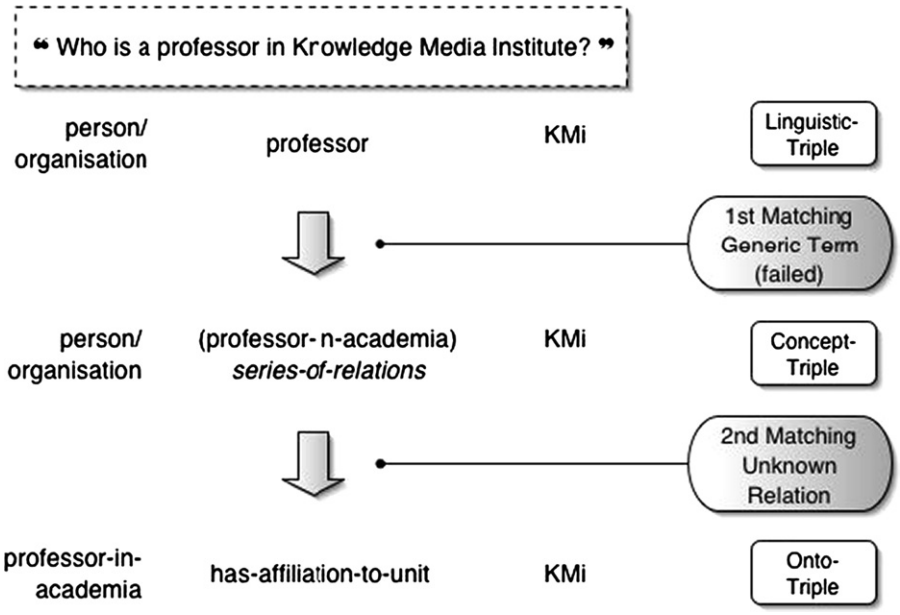

Fig. 19. Learning mechanism matching example.

the database query will return many more mappings and, quite likely, also meanings that are not desired.

Future work on the learning mechanism will investigate the augmentation of the user-profile. In fact, through a specific auxiliary ontology that describes a series of user's profiles, it would be possible to infer connections between the type of mapping and the type of user. Namely, it would be possible to correlate a particular jargon to a set of users. Moreover, through a reasoning service, this correlation could become dynamic, being continually extended or diminished consistently with the relations between user's choices and user's information. For example, if the LM detects that a number of registered users, all characterized as PhD students, keep employing the same jargon, it could extend the same mappings to all the other registered PhD students.

7. Evaluation

AquaLog allows users who have a question in mind and know something about the domain to query the semantic markup viewed as a knowledge base. The aim is to provide a system which does not require users to learn specialized vocabulary, or to know the structure of the knowledge base, but as pointed out in Ref. [14], though they have to have some idea of the contents of the domain they may have some misconceptions. Therefore, to be realistic, some process of familiarization at least is normally required.

A full evaluation of AquaLog requires both an evaluation of its query answering ability and the overall user experience (Section 7.2). Moreover, because one of our key aims is to make AquaLog an interface for the semantic web, the portability

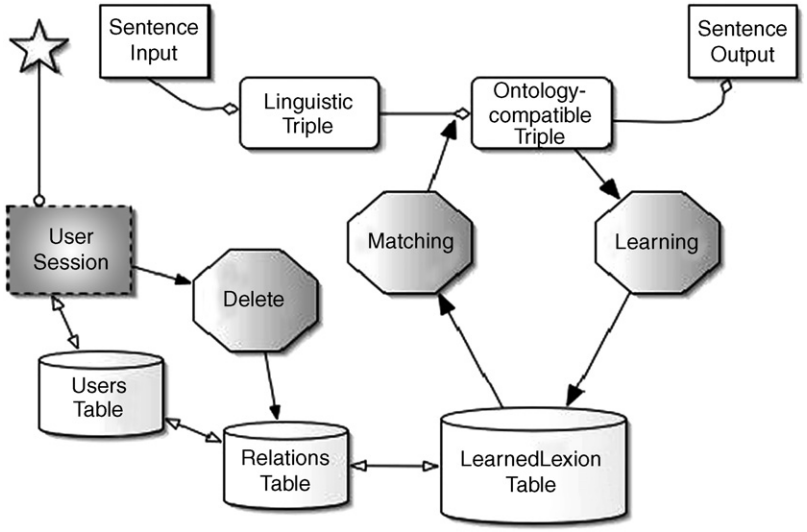

Fig. 20. Architecture to support a users' community.

across ontologies will also have to be addressed formally (Section 7.3).

We analyzed the failures and divided them into the following five categories (note that a query may fail at several different levels, e.g. linguistic failure and service failure, though conceptual failure is only computed when there is not any other kind of failure):

- *Linguistic failure*. This occurs when the NLP component is unable to generate the intermediate representation (but the question can usually be reformulated and answered).
- *Data model failure*. This occurs when the NL query is simply too complicated for the intermediate representation.
- *RSS failure*. This occurs when the Relation Similarity Service of AquaLog is unable to map an intermediate representation to the correct ontology-compliant logical expression.
- *Conceptual failure*. This occurs when the ontology does not cover the query, e.g. lack of an appropriate ontology relation or term to map with, or if the ontology is wrongly populated in a way that hampers the mapping (e.g., instances that should be classes).
- *Service failure*. In the context of the semantic web, we believe that these failures are less to do with shortcomings of the ontology than with the lack of appropriate services, defined over the ontology.

The improvement achieved due to the use of the Learning Mechanism (LM) in reducing the number of user interactions is also evaluated in Section 7.2. First, AquaLog is evaluated with the LM deactivated. For a second test, the LM is activated at the beginning of the experiment in which the database is empty. In the third test a second iteration is performed over the results obtained from the second iteration.

7.1. Correctness of an answer

Users query the information using terms from their own jargon. This NL query is formalized into predicates that correspond to classes and relations in the ontology. Further, variables in a query correspond to instances in that ontology. Therefore, an answer is judged as correct with respect to a query over a single specific ontology. In order for an answer to be correct, AquaLog has to align the vocabularies of both the asking query and the answering ontology. Therefore a valid answer is the one considered correct in the ontology world.

AquaLog fails to give an answer if the knowledge is in the ontology but it cannot find it (linguistic failure, data model failure, RSS failure or service failure). Please note that a conceptual failure is not considered as an AquaLog failure because the ontology does not cover the information needed to map all the terms or relations in the user query. Moreover, a correct answer, corresponding to a complete ontology-compliant representation, may give no results at all because the ontology is not populated.

AquaLog may need the user to help disambiguate a possible mapping for the query. This is not considered a failure. However, the performance of AquaLog was also evaluated for it. The user can decide to use the LM or not, that decision has a direct impact on the performance as the results will show.

7.2. Query answering ability

The aim is to assess to what extent the AquaLog application built using AquaLog with the AKT ontology[10] and the KMi knowledge base satisfied user expectations about the range of questions the system should be able to answer. The ontology used for the evaluation is also part of the KMi semantic portal.[11] Therefore, the knowledge base is dynamically populated and constantly changing, and as a result of this a valid answer to a query may be different at different moments.

This version of AquaLog does not try to handle a NL query if the query is not understood and classified into a type. It is quite easy to extend the linguistic component to increase AquaLog linguistic abilities. However, it is quite difficult to devise a sublanguage which is sufficiently expressive, therefore we also evaluate if a question can be easily reformulated to be understood by AquaLog. A second aim of the experiment was also to provide information about the nature of the possible extensions needed to the ontology and the linguistic components—i.e., we not only wanted to assess the current coverage of AquaLog but also to get some data about the complexity of the possible changes required to generate the next version of the system.

Thus, we asked 10 members of KMi, none of whom had been involved in the AquaLog project, to generate questions for the system. Because one of the aims of the experiment was to measure the linguistic coverage of AquaLog with respect to user needs, we did not provide them with much information about AquaLog's linguistic ability. However, we did tell them something about the conceptual coverage of the ontology, pointing out that its aim was to model the key elements of a research lab, such as publications, technologies, projects, research areas, people, etc. We also pointed out that the current KB is limited in its handling of temporal information; therefore we asked them not to ask questions which required temporal reasoning (e.g. today, last month, between 2004 and 2005, last year, etc.). Because no 'quality control' was carried out on the questions, it was admissible for these to contain some spelling mistakes and even grammatical errors. Also, we pointed out that AquaLog is not a conversational system. Each query is resolved on its own with no references to previous queries.

7.2.1. Conclusions and discussion

7.2.1.1. Results. We collected in total 69 questions (see results in Table 1), 40 of which were handled correctly by AquaLog, i.e., approximately 58% of the total. This was a pretty good result, considering that almost no linguistic restriction was imposed on the questions.

A closer analysis shows that 7 of the 69 queries, 10.14% of the total, present a conceptual failure. Therefore, only 47.82% of

[10] http://kmi.open.ac.uk/projects/akt/ref-onto/.
[11] http://semanticweb.kmi.open.ac.uk.

Table 1
AquaLog evaluation

AquaLog success in creating a valid Onto-Triple, or if it fails to complete the Onto-Triple is due to an ontology conceptual failure	
Total number of successful queries	(40 of 69) **58%**—from this 10 of 33 (**30.30%**) has no answer because ontology is not populated with data.
Total number of successful queries without conceptual failure	(33 of 69) **47.82%**
Total number of queries with only conceptual failure	(7 of 69) **10.14%**
AquaLog failures	
Total number failures (same query can fail for more than one reason)	(29 of 69) **42%**
RSS failure	(8 of 69) **11.59%**
Service failure	(10 of 69) **14.49%**
Data Model failure	(0 of 69) **0%**
Linguistic failure	(16 of 69) **23.18%**
AquaLog easily reformulated questions	
Total num. queries easily reformulated	(19 of 29) **65.5%** of failures
With conceptual failure	(7 of 19) **36.84%**
No conceptual failure but no populated	(6 of 19) **31.57%**

Bold values represent the final results (%).

the total (33 of 69 queries) reached the ontology mapping stage. Furthermore 30.30% of the correctly ontology mapped queries (10 of the 33 queries) cannot be answered because the ontology does not contain the required data (not populated).

Therefore, although AquaLog handles 58% of the queries, due to the lack of conceptual coverage in the ontology and how it is populated, for the user only 33.33% (23 of 69) will have a result (a non-empty answer). These limitations on coverage will not be obvious for the user, as a result, independently of whether a particular set of queries in answered or not, the system becomes unusable. On the other hand these limitations are easily overcome by extending and populating the ontology. We believe that the quality of ontologies will improve thanks to the Semantic Web scenario. Moreover, as said before, AquaLog is currently integrated with the KMi semantic web portal which automatically populates the ontology with information found in departmental databases and web sites. Furthermore, for the next generation of our QA system (explained in Section 8.5) we are aiming to use more than one ontology, so the limitations in one ontology can be overcome by another ontology.

7.2.1.2. Analysis of AquaLog failures. The identified AquaLog limitations are explained in Section 8. However, here we present the common failures we encountered during our evaluation. Following our classification:

- Linguistic failures accounted for 23.18% of the total number of questions (16 of 69). This was by far the most common problem. In fact, we expected this considering the few linguistic restrictions imposed on the evaluation compared to the complexity of natural language. Errors were due to (1) questions not recognized or classified, like when a multiple relation is used, e.g. in "what are the challenges and objectives [...]" (2) questions annotated wrongly, like tokens not recognized as a verb by GATE, e.g. "present results", and therefore relations annotated wrongly, or because of the use of parenthesis in the middle of the query or special names like "dot.kom", or numbers like a year, e.g. "in 1995".
- *Data model failure*. Intriguingly this type of failure never occurred, and our intuition is that this was the case not only because the relational model is actually a pretty good way to model queries but also because the ontology-driven nature of the exercise ensured that people only formulated queries that could in theory (if not in practice) be answered by reasoning about triples in the departmental KB.
- RSS failures accounted for 11.59% of the total number of questions (8 of 69). The RSS fails to correctly map an ontology compliant query mainly due to (1) the use of nominal compounds (e.g., terms that are a combination of two classes as "akt researchers"); (2) when a set of candidate relations is obtained through the study of the ontology, the relation is selected through syntactic matching between labels, through the use of string metrics and WordNet synonyms, not by the meaning, e.g. in "what is John Domingue working on?" we map into the triple ⟨*organization*, *works-for*, *john-domingue*⟩ instead of ⟨*research-area*, *have-interest*, *john-domingue*⟩, or in "how can I contact Vanessa?" due to the string algorithms "contact" is mapped to the relation "employment-contract-type" between a "research-staff- member" and "employment-contract-type"; (3) queries type "how long" are not implemented in this version; (4) non-recognizing concepts in the ontology when they are accompanied by a verb as part of the linguistic relation, like the class "publication" on the linguistic relation "have publications".
- *Service failures*. Several queries essentially asked for services to be defined over the ontologies. For instance, one query asked about "the top researchers", which requires a mechanism for ranking researchers in the lab—people could be ranked according to citation impact, formal status in the department, etc. Therefore we defined this additional category which accounted for 14.49% of failed questions in the total number of question (10 of 69). In this evaluation we identified the following key words that are not understood by this AquaLog version: main, most, proportion, most, new, same as, share same, other and different. No work has been done yet in relation to the services failures. Three kind of

Table 2
Learning mechanism performance

Number of queries (total 45)	Without the LM	LM first iteration (from scratch)	LM second iteration
0 iterations with user	37.77% (17 of 45)	40% (18 of 45)	64.44% (29 of 45)
1 iteration with user	35.55% (16 of 45)	35.55% (16 of 45)	35.55% (16 of 45)
2 iterations with user	20% (9 of 45)	20.5% (9 of 45)	0% (0 of 45)
3 iterations with user	6.66% (3 of 45)	6.66% (2 of 45)	0% (0 of 45)
	(43 iterations)	(40 iterations)	(16 iterations)

services have been identified: ranking, similarity/comparative and for proportions/percentages, which remains a future line of work for future versions of the system.

7.2.1.3. Reformulation of questions. As indicated in Refs. [14], it is difficult to devise a sublanguage which is sufficiently expressive yet avoids ambiguity and seems reasonably natural. The linguistic and services failures together account for the 37.67% failures (the total number of failures including the RSS failures is 42%). On many occasions questions can be easily reformulated by the user to avoid these failures so that the question can be recognized by AquaLog (linguistic failure), avoiding the use of nominal compounds (typical RSS failure) or avoiding unnecessary functional words like different, main, most of (service failure). In our evaluation 27.53% of the total questions (19 of 69) will be correctly handled by AquaLog by easily reformulating them, that means 65.51% of the failures (19 of 29) can be easily avoided. An example of a query that AquaLog is not able to handle or easily reformulate is "which main areas are corporately funded?"

To conclude, if we relax the evaluation restrictions allowing reformulation then 85.5% of our queries can be handled by AquaLog (independently of the occurrence of a conceptual failure).

7.2.1.4. Performance. As the results show (see Table 2 and Fig. 21), using the LM in the best of the cases we can improve from 37.77% of queries that can be answered in an automatic way to 64.44% queries. Queries that need one iteration with the user are quite frequent (35.55%) even if the LM is used. This is because in this version of AquaLog the LM is only used for relations not for terms (e.g. if the user asks for the term "Peter" the RSS will require him to select between "Peter Scott", "Peter Whalley" and among all the other "Peter's" present in the KB).

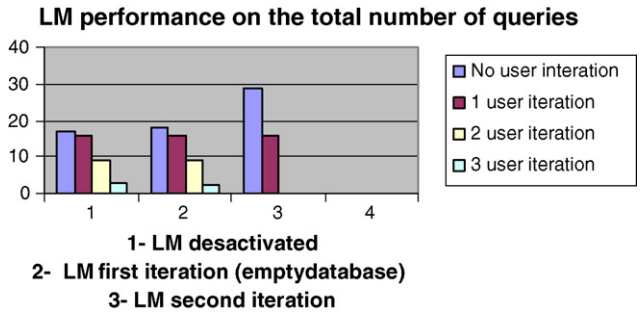

Fig. 21. Learning mechanism performance.

Furthermore, with the use of the LM the number of queries that need 2 or 3 iterations are dramatically reduced.

Note that the LM improves the performance of the system even for the first iteration (from 37.77% queries that require no iteration to 40%). Thanks to the use of the notion of context to find similar but not identical learned queries (explained in Section 6), the LM can help to disambiguate a query, even if it is the first time the query is presented to the system. These numbers can seem to the user like a small improvement in performance, however, we should take into account that the set of queries (total 45) is also quite small, logically the performance of the system for the 1st iteration of the LM improves if there is an increment in the number of queries.

7.3. Portability across ontologies

In order to evaluate portability we interfaced AquaLog to the Wine Ontology [50], an ontology used to illustrate the specification of the OWL W3C recommendation. The experiment confirmed the assumption that AquaLog is ontology portable, as we did not notice any hitch in the behaviour of this configuration compared to the others built previously. However, this ontology highlighted some AquaLog limitations to be addressed. For instance, a question like "which wines are recommended with cakes?" will fail because there is not a direct relation between wines and desserts; there is a mediating concept called "meal-course". However, the knowledge is in the ontology, and the question can be addressed if reformulated as "what wines are recommended for dessert courses based on cakes?"

The wine ontology is only an example for the OWL language, therefore it does not have much information instantiated, and, as a result, it does not present a complete coverage for many of our queries or no answer can be found for most of the questions. However, it is a good test case for the evaluation of the Linguistic and Similarity Components responsible for creating the correct translated ontology compliance triple (from which an answer can be inferred in a relatively easy way). More importantly, it makes evident what are the AquaLog assumptions over the ontology, as explained in the next subsection.

The "wine and food ontology" was translated into OCML and stored in the WebOnto server,[12] so that the AquaLog WebOnto plugin was used. For the configuration of AquaLog with the new ontology it was enough to modify the configuration files specifying the ontology server name and location and the ontology name. A quick familiarization with the ontology allowed us in

[12] http://plainmoor.open.ac.uk:3000/webonto.

the first instance to define the AquaLog ontology assumptions that are presented in the next subsection. In second instance, having a look at the few concepts in the ontology allowed us to configure the file in which "where" questions are associated with the concept "region" and "when" with the concept "vintageyear" (there is no appropriate association for "who" queries). In third instance, the lexicon can be manually augmented with synonyms for the ontology class "mealcourse", like "starter", "entrée", "food", "snack", "supper", or like "pudding" for the ontology class "dessertcourse". Though this last point it is not mandatory it improves the recall of the system, especially in cases where WordNet does not provide all frequent synonyms.

The questions used in this evaluation were based on the Wine Society "Wine and Food a Guideline" by Marcel Orford Williams,[13] as our own wine and food knowledge was not deep enough to make varied questions. They were formulated by a user familiar with AquaLog (the third author), as in contrast with the previous evaluation in which questions were drawn from a pool of users outside the AquaLog project. The reason for this is the different purpose in the evaluation, while in the first experiment one of the aims was the satisfaction of the user in respect to the linguistic coverage, in this second experiment we were interested in a software evaluation approach in which the tester is quite familiar with the system and can formulate a subset of complex questions intended to test the performance beyond the linguistic limitations (which can be extended by augmenting the grammars).

7.3.1. AquaLog assumptions over the ontology

In this section, we examine key assumptions that AquaLog makes about the format of semantic information it handles and how these balance portability against reasoning power.

In AquaLog we use triples as the intermediate representation (obtained from the NL query). These triples are mapped onto the ontology by a "light" reasoning about the ontology taxonomy, types, relationships and instances. In a portable ontology-based system for the open SW scenario we cannot expect complex reasoning over very expressive ontologies, because this requires detailed knowledge of ontology structure. A similar philosophy was applied to the design of the query interface used in the TAP framework for semantic searches [22]. This query interface, called GetData, was created to provide a lighter weight interface (in contrast to very expressive query languages that require lots of computational resources to process, such as the query languages SeRQL developed for to Sesame RDF platform and SPARQL for OWL). Guha et al. argue that "The lighter weight query language could be used for querying on the open, uncontrolled Web in contexts where the site might not have much control over who is issuing the queries, whereas a more complete query language interface is targeted at the comparatively better behaved and more predictable area behind the firewall" [22]. GetData provides a simple interface to data presented as directed labeled graphs, which does not assume prior knowledge of the ontological structure of the data. Similarly, AquaLog plugins provide an API (or ontology interface) that allows us to query and access the values of the ontology properties in the local or remote servers.

The "light-weight" semantic information imposes a few requirements on the ontology for AquaLog to get an answer. The ontology must be structured as a directed labeled graph (taxonomy and relationships) and the requirement to get an answer is that the ontology must be populated (triples) in such a way that there is a short (two relations or less), direct path between the query terms when mapped to the ontology.

We illustrate these limitations with an example using the wine ontology. We show how the requirements of AquaLog to obtain answers to questions about matching wines and foods compares to that of a purpose built agent programmed by an expert with full knowledge of the ontology structure. In the domain ontology wines are represented as individuals. *Mealcourses* are pairings of foods with drinks; their definition ends with restrictions on the properties of any drink that might be associated with such a course (Fig. 22). There are no instances of *mealcourses* linking specific wines with food in the ontology.

A system that has been build specifically for this ontology is the Wine Agent [26]. The Wine Agent interface offers a selection of types of course and individual foods as a mock-up menu. When the user has chosen a meal, the Wine Agent lists the properties of a suitable wine, in terms of its colour, sugar (sweet or dry), body, and flavor. The list of properties is used to generate an explanation of the inference process, and to search for a list of wines that match the desired characteristics.

In this way the Wine Agent can answer questions matching food and wine by exploiting domain knowledge about the role of *restrictions* which are a feature peculiar to that ontology. The method is elegant but is only portable to ontologies that use restrictions in the exact same way as the wine ontology does. In AquaLog, on the other hand, we were aiming to build a portable system targeted to the Semantic Web. Therefore, we chose to build an interface for querying ontology taxonomy, types, properties and instances, i.e. structures which are almost universal in Semantic Web ontologies. AquaLog cannot reason with ontology peculiarities, such as the restrictions in the wine ontology. For AquaLog to get an answer the ontology would have to be populated with *mealcourses* that do specify individual wines. In order to demonstrate this in the evaluation e introduced a new set of instances of *mealcourse* (e.g., see Table 3 for an example of instances). These provide direct paths, two triples in length, between the query terms that allow AquaLog to answer questions about matching foods to wines.

7.3.2. Conclusions and discussion

7.3.2.1. Results. We collected in total 68 questions. As the wine ontology is an example for the OWL language the ontology does

Table 3
Example of instances definition in OCML

(def-instance new-course7 (othertomatobasedfoodcourse ((hasfood pizza) (has drink mariettazinfadel)))	(def-instance new course8 mealcourse ((hasfood fowl) (hasdrink Riesling)))

[13] http://www.thewinesociety.com.

```
<OtherTomatoBasedFood rdf:ID="Pizza" />

<owl:Class rdf:ID="OtherTomatoBasedFoodCourse">

<rdfs:subClassOf> <owl:Restriction>
<owl:onProperty rdf:resource="#hasDrink" />
      <owl:allValuesFrom> <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/TR/2003/PR-owl-
      guide-20031209/wine#hasColor" />
      <owl:hasValue rdf:resource="#Red" />
      </owl:Restriction> </owl:allValuesFrom>
</owl:Restriction> </rdfs:subClassOf>

<rdfs:subClassOf> <owl:Restriction>
<owl:onProperty rdf:resource="#hasDrink" />
      <owl:allValuesFrom> <owl:Restriction>
      <owl:onProperty rdf:resource="http://www.w3.org/TR/2003/PR-owl-
      guide-20031209/wine#hasFlavor" />
      <owl:hasValue rdf:resource="#Moderate" />
      </owl:Restriction> </owl:allValuesFrom>
</owl:Restriction> </rdfs:subClassOf>
```

Fig. 22. OWL example of the wine and food ontology.

not present a complete coverage in terms of instances and terminology. Therefore, 51.47% of the queries fail because the information is not in the ontology. As previously, conceptual failures are only counted when there is not any other error. Following this we can consider them as correctly handled by AquaLog, though from the point of view of a user the system will not get an answer. AquaLog resolved 17.64% of questions (without conceptual failure though the ontology had to be populated by us to get an answer). Therefore, we can conclude that AquaLog was able to handle 51.47% + 17.64% = 69.11% of the total number of questions. If we allow the user to reformulate the questions the performance of AquaLog (independently of conceptual failures) improves to 89.70% of the total questions (66.66% of the failures are skipped by easily reformulating the query). All these results are presented in Table 4.

Due to the lack of conceptual coverage and the few relationships between concepts in this ontology, this is not an appropriate experiment to show the performance of the Learning Mechanism as very little interactivity from the user is needed.

Table 4
AquaLog evaluation

AquaLog success in creating a valid Onto-Triple, or if it fails to complete the Onto-Triple is due to an ontology conceptual failure	
Total number of successful queries	(47 of 68) **69.11%**
Total number of successful queries without conceptual failure	(12 of 68) **17.64%**
Total number of queries with only conceptual failure	(35 of 68) **51.47%**
AquaLog failures	
Total number failures (same query can fail for more than one reason)	(21 of 68) **30.88%**
RSS failure	(15 of 68) **22%**
Service failure	(0 of 68) **0%**
Data model failure	(0 0f 68) **0%**
Linguistic failure	(9 of 68) **13.23%**
AquaLog easily reformulated questions	
Total num. queries easily reformulated	(14 of 21) **66.66%** of failures
With conceptual failure	(9 of 14) **64.28%**

Bold values represent the final results (%).

7.3.2.2. Analysis of AquaLog failures. The identified AquaLog limitations are explained in Section 8. However, here we present the common failures we encountered during our evaluation. Following our classification:

- Linguistic failures accounted for 13.23% (9 of 68). All the linguistic failures were due to only two reasons. First reason is tokens that were not correctly annotated by GATE, e.g., not recognizing "asparagus" as a noun, or misunderstanding "smoked" as a verb/relation instead of as part of a name in "smoked salmon", similar situations occurred with terms like "grilled swordfish" or "cured ham". The second cause for failure is that it does not classify queries formed with "like" or "such as", e.g. "what goes well with a cheese like brie?" These kind of linguistic failures are easily overcome by extending the set of JAPE grammars and QA abilities.
- Data model failure accounted for 0% (0 of 68). As in the previous evaluation this type of failure never occurred. All questions can be easily formulated as AquaLog triples.
- RSS failures accounted for 22% (15 of 68). This being the most common problem. The structure of this ontology, with only a few concepts but strongly based on the use of mediating concepts and few direct relationships, highlighted the main RSS limitations explained in Section 8. These interesting limitations would have to be addressed in the next AquaLog generation. Interestingly, all these RSS failures can be skipped by reformulating the query, they are further explained in the next section "Similarity failures and reformulation of questions".
- Service failures accounted for 0% (0 of 69). This type of failure never occurs. This was the case because the user who generated the questions was familiar enough with the system and the user did not ask questions that require ranking or comparative services.

7.3.2.3. Similarity failures and reformulation of questions. All the questions that fail due to only RSS shortcomings can be easily reformulated by the user into a question in which the RSS

can generate a correct ontology mapping. That is 66.66% of the total erroneous questions can be reformulated, allowing this AquaLog version to be able to correctly handle 89.70% of the queries.

However, this ontology highlighted the main AquaLog limitations discussed in Section 8. This provides us valuable information about the nature of possible extensions needed on the RSS components. We did not only want to assess the current coverage of AquaLog but also to get some data about the complexity of the possible changes required to generate the next version of the system.

The main underlying reason for most of the RSS failures is the structure of the wine and food ontology strongly based on using mediating concepts instead of direct relationships. For instance, the concept "meal" is related to "mealcourse" through an "ad hoc" relation. "mealcourse" is the mediating concept between "meal" and all kinds of food: ⟨*meal*, *course*, *mealcourse*⟩, ⟨*mealcourse*, *has-food*, *nonredmeat*⟩. Moreover, the concept "wine" is related to food only thought the concept "mealcourse" and its subclasses (e.g. "dessert course"), ⟨*mealcourse*, *has-drink*, *wine*⟩. Therefore, queries like "which wines should be served with a meal based on pork?" should be reformulated to "which wines should be served with a mealcourse based on pork?" to be answered. Same applies for the concepts "dessert" and "dessert course", for instance if we formulate the query "what can I serve with a dessert of pie?" AquaLog wrongly generates the ontology triples ⟨*which is*, *madefromfruit*, *dessert*⟩ and ⟨*dessert*, *?*, *pie*⟩ it fails to find the correct relation for "dessert" because it is taking the direct relation "madefromfruit" instead of going through the mediating concept "mealcourse". Moreover, for the second triple it fails to find an "ad hoc" relation between "apple pie" and "dessert" because "pie" is a subclass of "dessert". The question is correctly answered when reformulated to "what should I serve with a dessert course of apple pie?" As explained in Section 8, for the next AquaLog generation the RSS must be able to reclassify the triples and dynamically generate a new triple to map indirect relationships (through a mediating concept) if needed.

Other minor RSS failures are due to nominal compounds (as in the previous evaluation), e.g. "wine regions", or for instance, in the basic generic query "what should we drink with a starter of seafood?" AquaLog is trying to map the term "seafood" into an instance, however "seafood" is a class in the food ontology. This case should be contemplated in the next AquaLog version.

8. Discussion, limitations and future lines

We examined the nature of the AquaLog question answering task itself and identified some major problems that we need to address when creating a NL interface to ontologies. Limitations are due to linguistic and semantic coverage, lack of appropriate reasoning services or lack of enough mapping mechanisms to interpret a query, and the constraints imposed by ontology structures.

8.1. Question types

The first limitation of AquaLog related to the type of questions being handled. We can distinguish different kind of questions: affirmative/negative type of questions, "wh" questions (who, what, how many...) and commands (Name all...). All of these are treated as questions in AquaLog. As pointed out in Ref. [24] there is evidence that some kinds of questions are harder than others when querying *free text*. For example, "why" and "how" questions tend to be difficult, because they require understanding of causality or instrumental relations. "What" questions are hard, because they provide little constraint on the answer type. By contrast, AquaLog can handle "what" questions easily as the possible answer types are constrained by the types of the possible relationships in the ontology.

AquaLog only supports questions referring to the present state of the world, as represented by an ontology. It has been designed to interface ontologies that facilitate little time dependent information and has limited capability to reason about temporal issues. Although simple questions formed with "how-long" or "when", like "when did the akt project start?" can be handled, it cannot cope with expressions like "in the last year". This is because the structure of temporal data is domain dependent; compare geological time scales to the periods of time in a knowledge base about research projects. There is a serious research challenge in determining how to handle temporal data in a way which would be portable across ontologies.

The current version also does not understand queries formed with "how much". This is mainly because the Linguistic Component does not yet fully exploit quantifier scoping [18] ("each", "all", "some"). Unlike temporal data, our intuition is that some basic aspects of quantification are domain independent and we are working on improving this facility.

In short, the limitations on the types of question that can be asked of AquaLog are imposed, in large part, by limitations on the kinds of generic query methods that are portable across ontologies. The use of ontologies extends its ability to ask "what" and "how" questions, but the implementation of temporal structures and causality ("why" and "how" questions) is ontology dependent so these features could not be built into AquaLog without sacrificing portability.

There are some linguistic forms that it would be helpful for AquaLog to handle but which we have not yet tackled. These include other languages, genitive ('s) and negative words (such as "not" and "never" or implicit negatives such as "only", "except" and "other than"). AquaLog should also include a count feature in the triples indicating the number of instances to be retrieved, to answer queries like "Name 30 different people..."

There are also some linguistic forms that we do not believe it is worth concentrating on. Since AquaLog is used for standalone questions, it does not need to handle the linguistic phenomenon called anaphora, in which pronouns (e.g. "she", "they"), and possessive determiners (e.g. "his", "theirs") are used to denote implicitly entities mentioned in an extended discourse. However, some kinds of noun phrases which occur beyond the same sentence are understood, i.e. "is there any [...] who/that/which [...]"

8.2. Question complexity

Question complexity also influences the performance of QA. The system described in Ref. [36], DIMAP, has in average 3.3 triples per questions. However, in Ref. [36] triples are formed in a different way. AquaLog has a triple for relationship between terms, even if the relation is not explicit. DIMAP has a triple for discourse entity (for each unbound variable) in which a semantic relation is not strictly required. At a later stage DIMAP triples are consolidating so that contiguous entities are made into single triple: "questions containing the phrase "who was the author" were converting into "who wrote". That pre-processing is done by the AquaLog linguistic component so that it can obtain the minimum required set of Query-Triples to represent a NL query at once, independently of how this was formulated.

On the other hand, AquaLog can only deal with questions which generate a maximum of two triples. Most of the questions we encountered in the evaluation study were not complex but we have identified a few cases which require more than two triples. The triples are fixed for each query category, but we found examples in which the numbers of triples is not obvious at the first stage and may change to adapt to the semantics of the ontology (dynamic creation of triples by the RSS). For instance, when dealing with compound nouns: for a term like "French researchers" a new triple must be created when the RSS detects during the semantic analysis phase that it is in fact a compound noun, as there is an implicit relationship between French and researchers. The compound noun problem is discussed further below.

Another example is in the question "which researchers have publications in KMi related to social aspects", where the linguistic triples obtained are: ⟨researchers, have publications, KMi⟩ ⟨which is/are, related, social aspects⟩. However, the relation "have publications" refers to "publication: has author" in the ontology. Therefore, we need to represent three relationships between the terms: ⟨research, ?, publications⟩ ⟨publications, ?, KMi⟩ ⟨publications, related, social aspects⟩, therefore three triples instead of two are needed.

Along the same lines, we have observed cases where terms need to be bridged by multiple triples of unknown type. AquaLog cannot at the moment associate linguistic relations to non-atomic semantic relations. In other words, it cannot infer an answer if there is not a direct relation in the ontology between the two terms implied in the relationship, even if the answer is in the ontology. For example, imagine the question "who collaborates with IBM?", where there is not a "collaboration" relation between a person and an organization, but there is a relation between a "person" and a "grant" and a "grant" with an "organization". The answer is not a direct relation and the graph in the ontology can only be found by expanding the query with the additional term "grant" (see also the "mealcourse" example in Section 7.3 using the wine ontology).

With the complexity of questions, as with their type, the ontology design determines the questions which may be successfully answered. For example, when one of the terms in the query is not represented as an instance, relation or concept in the ontology but as a value of a relation (string). Consider the question "who is the director in KMi?" In the ontology, it may be represented as "Enrico Motta – has job title – KMi director", where "KMi director" is a String. AquaLog is not able to find it as it is not possible to check in real time all the string values for each instance in the ontology. The information is only accessible if the question is reformulated to "what is the job title of Enrico?", so we would get "KMi-director" as an answer.

AquaLog has not yet solved the nominal compound problem identified in Ref. [3]. In English nouns are often modified by other preceding nouns (i.e. "semantic web papers", "research department"). A mechanism must be implemented to identify whether a term is in fact a combination of terms which are not separated by a preposition. Similar difficulties arise in the case of adjective-noun compounds. For example, a "large company" may be a company with a large volume of sales or a company with many employees [3]. Some systems require the meaning of each possible noun-noun or adjective-noun compound to be declared during the configuration phase. The configurer is asked to define the meaning of each possible compound in terms of concepts of the underlying domain [3].

8.3. Linguistic ambiguities

The current version of AquaLog has restricted facilities for detecting and dealing with questions which are ambiguous.

The role of logical connectives is not fully exploited. These have been recognized as a potential source of ambiguity in question answering. Androutsopoulos et al. [3], presents the example "list all applicants who live in California and Arizona". In this case the word "and" can be used to denote either disjunction or conjunction, introducing ambiguities which are difficult to resolve. (In fact, the possibility for "and" to denote a conjunction here should be ruled out, since an applicant cannot normally live in more than one state, but this requires domain knowledge.) In the next AquaLog version, either a heuristic rule must be implemented to select disjunction or conjunction or an answer for both must be given by the system.

An additional linguistic feature not exploited by the RSS is the use of the person and number of the verb to resolve the attachment of subordinate clauses. For example, consider the difference between the sentences "which *academic* works with peter who has interests in the semantic web?" and "which *academics* work with peter who has interests in the semantic web?" The former example is truly ambiguous—either "Peter" or the "academic" could have an interest in the semantic web. However the latter can be solved if we take into account that the verb "has" is in the third person singular, therefore the second part "has interests in the semantic web" must be linked to "Peter" for the sentence to be syntactically correct. This approach is not implemented in AquaLog. The obvious disadvantage for AquaLog is that user's interaction will be required in both examples. The advantage is that some robustness is obtained over syntactical incorrect sentences. Whereas methods such as the person and number of the verb assume that all sentences are well-formed. Our experience with the sample of questions we gathered for the evaluation study was that this is not necessarily the case.

8.4. Reasoning mechanisms and services

A future AquaLog version should include Ranking Services that will solve questions like "what are the most successful projects?" or "what are the five key projects in KMi?" To answer these questions, the system must be able to carry out reasoning based on the information present in the ontology. These services must be able to manipulate both general and ontology-dependent knowledge, as for instance, the criteria which make a project successful could be the number of citations or the amount funding. Therefore, the first problem identified is how can we make these criteria as ontology independent as possible, and available to any application that may need them. An example of deductive components with an axiomatic application domain theory to infer an answer can be found in Ref. [51].

Alternatively, the learning mechanism can be extended to learn typical services, mentioned above, that people require when posing queries to a semantic web. Those services are not supported by domain relations—they are essentially meta-level services. These meta-relations can be defined by providing a mechanism that allows the user to augment the system by manual association of meta-relations to domain relations.

For example, if the question "what are the cool projects for a PhD student?" is asked, the system would not be able to solve the linguistic ambiguity of the words "cool project". The first time, some help from the user is needed, who may select "all projects which belong to a research area in which there are many publications". A mapping is therefore created between "cool projects", "research area" and "publications", so that the next time the Learning Mechanism is called it will be able to recognize this specific user jargon.

However, the notion of communities is fundamental in order to deliver a feasible substitution service. In fact, another person could use the same jargon, but with another meaning. Let's imagine a professor, who asks the system a similar question "what are the cool projects for a professor?" What he means by saying "cool projects" is "funding opportunity". Therefore, the two mappings entail two different contexts depending on the users' community.

We also need fallback options to provide some answer when more intelligent mechanisms fail. For example, when a question is not classified by the Linguistic Component the system could look for an ontology path between the terms recognized in the sentence. This might tell the user about projects that are currently associated with professors. This may help the user reformulate the question about "coolness" in a way the system can support.

8.5. New research lines

While the current version of AquaLog is portable from one domain to the other (the system being agnostic to the domain of the underlying ontology), the scope of the system is limited by the amount of knowledge encoded in the ontology. One way to overcome this limitation is to extend AquaLog in the direction of open question answering, i.e., allowing the system to combine knowledge from the wide range of ontologies autonomously created and maintained on the semantic web. This new re-implementation of AquaLog, currently under development, is called PowerAqua [37]. In a heterogeneous, open domain scenario it is not possible to determine in advance which ontologies will be relevant to a particular query. Moreover, it is often the case that queries can only be solved by composing information derived from multiple and autonomous information sources. Hence, to perform QA we need a system which is able to locate and aggregate information in real time, without making any assumptions about the ontological structure of the relevant information.

AquaLog bridges between the terminology used by the user and the concepts used in the underlying ontology. PowerAqua will function in the same way as AquaLog does, with the essential difference that the terms of the question will need to be dynamically mapped to *several* ontologies. AquaLog's linguistic component and RSS algorithms to handle ambiguity within one ontology, in particular regarding modifier attachment, will be fully reused. Moreover, in both AquaLog and PowerAqua there is no pre-determined assumption of the ontological structure required for complex reasoning, therefore the AquaLog evaluation and analysis presented here, highlighted many of the issues to take into account when building an ontology-based QA.

However, building a multi-ontology QA system, in which portability is no longer enough and "openness" is required, brings up new challenges in comparison with AquaLog that have to be solved by PowerAqua in order to interpret a query by means of different ontologies. These various issues are: resource selection, heterogeneous vocabularies and combination of knowledge from different sources. Firstly, we must address the automatic selection of the potentially relevant KB/ontologies to answer a query. In the SW we envision a scenario where the user has to interact with several large ontologies, therefore indexing may be needed to perform searches at run time. Secondly, user terminology may be translated into semantically sound triples containing terminology distributed across ontologies; in any strategy that focuses on information content, the most critical problem is that of different vocabularies used to describe similar information across domains. Finally, queries may have to be answered not by a single knowledge source but by consulting multiple sources, and therefore, combining the relevant information from different repositories to generate a complete ontological translation for the user queries and identifying common objects. Among other things, this requires the ability to recognize whether two instances coming from different sources may actually refer to the same individual.

9. Related work

QA systems attempt to allow users to ask a query in NL and receive a concise answer, possibly with validating context [24]. AquaLog is an ontology-based NL QA system to query the semantic mark-up. The novelty of the system with respect to traditional QA systems is that it relies on the knowledge encoded in the underlying ontology and its explicit semantics to disambiguate the meaning of the questions and provide answers, and as pointed on the first AquaLog conference publication [38] it provides a new 'twist' on the old issues of NLIDB. To see to

what extent AquaLog's novel approach facilitates the QA processing and presents advantages with respect to NL interfaces to databases and open domain QA we focus on the following: (1) portability across domains; (2) dealing with unknown vocabulary; (3) solving ambiguity; (4) supported question types; (5) supported question complexity.

9.1. Closed-domain natural language interfaces

This scenario is of course very similar to asking natural language queries to databases (NLIDB), which has long been an area of research in the artificial intelligence and database communities even if in the past decade it has somewhat gone out of fashion [3]. However, it is our view that the SW provides a new and potentially important context in which the results from this area can be applied. The use of natural language to access relational databases can be traced back to the late sixties and early seventies [3].[14] In Ref. [3] a detailed overview of the state of the art for these systems can be found.

Most of the early NLIDBs systems were built having a particular database in mind, thus they could not be easily modified to be used with different databases or were difficult to port to different application domains (different grammars, hard-wired knowledge or mapping rules had to be developed). Configuration phases are tedious and require a long time. AquaLog portability costs, instead, are almost zero. Some of the early NLIDBs relied on pattern-matching techniques. In the example described by Androutsopoulos in Ref. [3], a rule says that if a user's request contains the word "capital" followed by a country name, the system should print the capital which corresponds to the country name, so the same rule will handle "what is the capital of Italy?", "print the capital of Italy", "Could you please tell me the capital of Italy". The shallowness of the pattern-matching would often lead to bad failures. However, a domain independent analog of this approach may be used in the next AquaLog version as a fallback mechanism whenever AquaLog is not able to understand a query. For example, if it does not recognize the structure "Could you please tell me", so instead of giving a failure, it could get the terms "capital" and "Italy", create a triple with them, and, using the semantics offered by the ontology, look for a path between them.

Other approaches are based on statistical or semantic similarity. For example, FAQ Finder [9] is a natural language QA system that uses files of FAQs as its knowledge base; it also uses WordNet to improve its ability to match questions to answers, using two metrics: statistical similarity and semantic similarity. However, the statistical approach is unlikely to be useful to us because it is generally accepted that only long documents with large quantities of data have enough words for statistical comparisons to be considered meaningful [9], which is not the case when using ontologies and KBs instead of text. Semantic similarity scores rely on finding connections through WordNet between the user's question and the answer. The main problem here is the inability to cope with words that apparently are not found in the KB.

The next generation of NLIDBs used an intermediate representation language, which expressed the meaning of the user's question in terms of high-level concepts, independent of the database structure [3]. For instance, the approach of the NL system for databases based on formal semantics presented in Ref. [17] made a clear separation between the NL front ends, which have a very high degree of portability, and the back end. The front end provides a mapping between sentences of English and expressions of a formal semantic theory, and the back end maps these into expressions which are meaningful with respect to the domain in question. Adapting a developed system to a new application will only involve altering the domain specific back end. The main difference between AquaLog and the latest generation of NLIDB systems [17] is that AquaLog uses an intermediate representation through the entire process, from the representation of the user's query (NL front end) to the representation of an ontology compliant triple (through similarity services), from which an answer can be directly inferred. It takes advantage of the use of ontologies and generic resources in a way that makes the entire process highly portable.

TEAM [39] is an experimental, transportable NLIDB developed in the 1980s. The TEAM QA system, like AquaLog, consists of two major components: (1) for mapping NL expressions into formal representations; (2) for transforming these representations into statements of a database, making a separation of the linguistic process and the mapping process onto the KB. To improve portability, TEAM requires "separation of domain-dependent knowledge (to be acquired for each new database) from the domain-independent parts of the system". AquaLog presents an elegant solution in which the domain-dependent knowledge is obtained through a learning mechanism in an automatic way. Also, all the domain dependent configuration parameters, like "who" is the same as "person", are specified in a XML file. In TEAM the logical form constructed constitutes an unambiguous representation of the English query. In AquaLog NL ambiguity is taken into account, so if the linguistic phase is not able to disambiguate it, the ambiguity goes into the next phase, the relation similarity service, which is an interactive service that tries to disambiguate using the semantics in the ontology or otherwise by asking for user feedback.

MASQUE/SQL [2] is a portable NL front-end to SQL databases. The semi-automatic configuration procedure uses a built-in domain editor which helps the user to describe the entity types to which the database refers, using an is-a hierarchy, and then to declare the words expected to appear in the NL questions and to define their meaning in terms of a logic predicate that is linked to a database table/view. In contrast with MASQUE/SQL, AquaLog uses the ontology to describe the entities with no need for an intensive configuration procedure.

More recent work in the area can be found in Ref. [46]. PRECISE [46] maps questions to the corresponding SQL query by identifying classes of questions that are easy to understand in a well defined sense: the paper defines a formal notion of semantically tractable questions. Questions are sets of attribute/value

[14] Example of such systems are: LUNAR, RENDEZVOUS, LADDER, CHAT-80, TEAM, ASK, JANUS, INTELLECT, BBn's PARLANCE, IBM's LANGUAGEACCESS, Q&A Symantec, NATURAL LANGUAGE/DATATALKER, LOQUI, ENGLISH WIZARD, MASQUE.

pairs and a relation token corresponds to either an attribute token or a value token. Each attribute in the database is associated with a wh-value (what, where, etc.). In PRECISE, like in AquaLog, a lexicon is used to find synonyms. However, in PRECISE the problem of finding a mapping from the tokenization to the database requires that all tokens must be distinct; questions with unknown words are not semantically tractable and cannot be handled. In other words, PRECISE will not answer a question that contains words absent from its lexicon. In contrast with PRECISE, AquaLog employs similarity services to interpret the user's query by means of the vocabulary in the ontology. As a consequence, AquaLog is able to reason about the ontology structure in order to make sense of unknown relations or classes which appear not to have any match in the KB or ontology. Using the example suggested in Ref. [46], the question "what are some of the neighborhoods of Chicago?" cannot be handled by PRECISE because the word "neighborhood" is unknown. However, AquaLog would not necessarily know the term "neighborhood", but it might know that it must look for the value of a relation defined for cities. In many cases this information is all AquaLog needs to interpret the query.

9.2. Open-domain QA systems

We have already pointed out that research in NLIDB is currently a bit 'dormant', therefore it is not surprising that most current work on QA, which has been rekindled largely by the Text Retrieval Conference[15] (see examples below), is somewhat different in nature from AquaLog. However, there are linguistic problems common in most kinds of NL understanding systems.

Question answering applications for text typically involve two steps, as cited by Hirschman [24]: (1) "Identifying the semantic type of the entity sought by the question"; (2) "Determining additional constraints on the answer entity". Constraints can include, for example, keywords (that may be expanded using synonyms or morphological variants) to be used in matching candidate answers; and syntactic or semantic relations between a candidate answer entity and other entities in the question. Various systems have, therefore built hierarchies of question types based on the types of answers sought [43,48,25,53].

For instance, in LASSO [43] a question type hierarchy was constructed from the analysis of the TREC-8 training data. Given a question, it can find automatically (a) the type of question (what, why, who, how, where), (b) the type of answer (person, location. . .), (c) the question focus, defined as the "main information required by the interrogation" (very useful for "what" questions which say nothing about the information asked for by the question). Furthermore, it identifies the keywords from the question. Occasionally, some words of the question do not occur in the answer (for example, if the focus is "day of the week" it is very unlikely to appear in the answer). Therefore, it implements similar heuristics to the ones used by named entity recognizer systems for locating the possible answers.

[15] sponsored by the American National Institute (NIST) and the Defense Advanced Research Projects Agency (DARPA). TREC introduces and open-domain QA track in 1999 (TREC-8).

Named entity recognition, and information extraction (IE) are powerful tools in question answering. One study showed that over 80% of questions asked for a named entity as a response [48]. In that work, Srihari and Li argue that:

> "(i) IE can provide solid support for QA; (ii) Low-level IE is often a necessary component in handling many types of questions; (iii) A robust natural language shallow parser provides a structural basis for handling questions; (iv) High-level domain independent IE is expected to bring about a breakthrough in QA."

Where point (iv) refers to the extraction of multiple relationships between entities and general event information like WHO did WHAT. AquaLog also subscribes to point (iii), however the main two differences between open-domains systems and ours are: (1) it is not necessary to build hierarchies, or heuristics to recognize named entities, as all the semantic information needed is in the ontology, (2) AquaLog has already implemented mechanisms to extract and exploit the relationships to understand a query. Nevertheless, the goal of the main similarity service in AquaLog, the RSS, is to map the relationships in the linguistic triple into an ontology-compliant-triple. As described in Ref. [48], NE is necessary but not complete in answering questions because NE by nature only extracts isolated individual entities from text, therefore methods like "the nearest NE to the queried key words" [48] are used.

Both AquaLog and open-domain systems attempt to find synonyms, plus their morphological variants, for the terms or keywords. Also in both cases, at times, the rules keep ambiguity unresolved and produce non-deterministic output for the asking point (for instance, "who" can be related to a "person" or to an "organization").

As in open-domain systems, AquaLog also automatically classifies the question before hand. The main difference is that AquaLog classifies the question based on the kind of triple which is a semantic equivalent representation of the question, while most of the open-domain QA systems classify questions according to their answer target. For example, in Wu et al.'s ILQUA system [53] these are categories like person, location, date, region or subcategories like lake, river. In AquaLog the triple contains information not only about the answer expected or focus, which is what we call the generic term or ground element of the triple, but also about the relationships between the generic term and the other terms participating in the question (each relationship is represented in a different triple). Different queries, formed by "what is", "who", "where", "tell me", etc., may belong to the same triple category, as they can be different ways of asking the same thing. An efficient system should therefore group together equivalent question types [25] independently of how the query is formulated, e.g. a basic generic triple represents a relationship between a ground term and an instance, in which the expected answer is a list of elements of the type of the ground term that occur in the relationship.

The best results of the TREC9 [16] competition were obtained by the FALCON system described in Harabagiu et al. [23]. In FALCON the answer semantic categories are mapped into categories covered by a Named Entity Recognizer. When the

question concept indicating the answer type is identified, it is mapped into an answer taxonomy. The top categories are connected to several word classes from WordNet. In an example presented in [23], FALCON identifies the expected answer type of the question "what do penguins eat?" as food because "it is the most widely used concept in the glosses of the subhierarchy of the noun synset {eating, feeding}". All nouns (and lexical alterations) immediately related to the concept that determines the answer type are considered among the keywords. Also, FALCON gives a cached answer if the similar question has already been asked before; a similarity measure is calculated to see if the given question is a reformulation of a previous one. A similar approach is adopted by the learning mechanism in AquaLog, where the similarity is given by the context stored in the triple.

Semantic web searches face the same problems as open domain systems in regards to dealing with heterogeneous data sources on the Semantic Web. Guha et al. [22] argue that "in the Semantic Web it is impossible to control what kind of data is available about any given object [. . .]. Here, there is a need to balance the variation in the data availability by making sure that at a minimum, certain properties are available for all objects. In particular data about the kind of object and how is referred to, i.e., rdf:type and rdfs:label". Similarly AquaLog makes simple assumptions about the data, which is understood as instances or concepts belonging to a taxonomy and having relationships with each other. In the TAP system [22] search is augmenting with data from the SW. To determine the concept denoted by the search query the first step is to map the search term to one or more nodes of the SW (this might return no candidate denotations, a single match or multiple matches). A term is searched by using its rdfs:label or one of the other properties indexed by the search interface. In ambiguous cases it chooses based on the popularity of the term (frequency of occurrence in a text corpus, the user profile, the search context) or by letting the user pick the right denotation. The node which is the selected denotation of the search term provides a starting point. Triples in the vicinity of each of these nodes are collected. As Ref. [22] argues:

> "the intuition behind this approach is that proximity in the graph reflects mutual relevance between nodes. This approach has the advantage of not requiring any hand-coding but has the disadvantage of being very sensitive to the representational choices made by the source on the SW [. . .]. A different approach is to manually specify for each class object of interest, the set of properties that should be gathered. A hybrid approach has most of the benefits of both approaches".

In AquaLog the vocabulary problem is solved (ontology triples mapping) not only by looking at the labels but also by looking at the lexically related words and the ontology (context of the triple: terms and relationships), and in the last resort, ambiguity is solved by asking the user.

9.3. Open-domain QA systems using triple representations

Other NL search engines such as AskJeeves [4] and Easy Ask [20] exist, which provide NL question interfaces to the Web but retrieve documents, not answers. AskJeeves relies on human editors to match question templates with authoritative sites; systems such as START [31], REXTOR [32] and AnswerBus [54], whose goal is also to extract answers from text.

START focuses on questions about geography and the MIT infolab. AquaLog's relational data model (triple-based) is somehow similar to the approach adopted by START called "object-property-value". The difference is that instead of properties we are looking for relations between terms, or between a term and its value. Using an example presented in Ref. [31]: "what languages are spoken in Guernsey?", for START the property is "languages" between the Object "Guernsey" and the value "French"; for AquaLog it will be translated into a relation "are spoken" between a term "language" and a location "Guernsey".

The system described in Litkowski [36], called DIMAP, extracts "semantic relation triples" after the document is parsed and the parse tree is examined. The DIMAP triples are stored in a database in order to be used to answer the question. The semantic relation triple described consists of a discourse entity (SUBJ, OBJ, TIME, NUM, ADJMOD), a semantic relation that "characterizes the entity's role in the sentence" and a governing word which is "the word in the sentence that the discourse entity stood in relation to". The parsing process generated an average of 9.8 triples per sentence. The same analysis was for each question, generating on average 3.3 triples per sentence, with one triple for each question containing an unbound variable, corresponding to the type of question. DIMAP-QA converts the document into triples. AquaLog uses the ontology, which may be seen as a collection of triples. One of the current AquaLog limitations is that the number of triples is fixed for each query category, although, the AquaLog triples change during its life cycle. However, the performance is still high as most of the questions can be translated into one or two triples. Apart from that the AquaLog triple is very similar to the DIMAP semantic relation triples. AquaLog has a triple for relationship between terms, even if the relationship is not explicit. DIMAP has a triple for discourse entity. A triple is generally equivalent to a logical form (where the operator is the semantic relation though is not strictly required). The discourse entities are the driving force in DIMAP triples, key elements (key nouns, key verbs, and any adjective or modifier noun) are determined for each question type. The system categorized questions in six types: time, location, who, what, size and number questions. In AquaLog, the discourse entity or the unbound variable can be equivalent to any of the concepts in the ontology (it does not affect the category of the triple), the category of the triple being the driving force, which determines the type of processing.

PiQASso [5] uses a "coarse-grained question taxonomy" consisting of person, organization, time, quantity and location as basic types plus 23 WordNet top-level noun categories. The answer type can combine categories. For example, in questions made with who, where the answer type is a person or an organization. Categories can often be determined directly from a wh-word: "who", "when", "where". In other cases additional information is needed. For example with "how" questions the

category is found from the adjective following "how" ("how many" or "how much") for quantity, "how long" or "how old" for time, etc. The type of "what ⟨noun⟩" question is normally the semantic type of the noun which is determined by WNSense, a tool for classifying word senses. Questions of the form "what ⟨verb⟩" have the same answer type as the object of the verb. "What is" questions in PiQASso also rely on finding the semantic type of a query word. However, as the authors say "it is often not possible to just look up the semantic type of the word, because lack of context does not allow identifying (*sic*) the right sense". Therefore, they accept entities of any type as answers to definition questions, provided they appear as the subject in an "is-a" sentence. If the answer type can be determined for a sentence it is submitted to the relation matching filter during this analysis, the parser tree is flattened into a set of triples and certain relations can be made explicit by adding links to the parser tree. For instance in Ref. [5] the example "man first walked on the moon in 1969" is presented, in which "1969" depends on "in", which in turn depends on "moon". Attardi et al. propose short circuiting the "in" by adding a direct link between "moon" and "1969" following a rule that says that "whenever two relevant nodes are linked through an irrelevant one, a link is added between them".

9.4. Ontologies in question answering

We have already mentioned that many systems simply use an ontology as a mechanism to support query expansion in information retrieval. In contrast with these systems, AquaLog is interested in providing answers, derived from semantic annotations, to queries expressed in NL. In the paper by Basili et al. [6], the possibility of building an ontology-based question answering system in the context of the semantic web is discussed. Their approach is being investigated in the context of EU project MOSES, with the "explicit objective of developing an ontology-based methodology to search, create, maintain and adapt semantically structured Web contents according to the vision of semantic web". As part of this project, they plan to investigate whether and how an ontological approach could support QA across sites. They have introduced a classification of the questions that the system is expected to support and see the content of a question largely in terms of concepts and relations from the ontology. Therefore the approach and scenario has many similarities with AquaLog. However, AquaLog is an implemented on-line system with wider linguistic coverage. The query classification is guided by the equivalent semantic representations or triples. The mapping process is converting the elements of the triple into entry-points to the ontology and KB. Also, there is a clear differentiation between the linguistic component and the ontology-base relation similarity services. The goal of the next generation of AquaLog is to use all available ontologies on the SW to produce an answer to a query, as explained in Section 8.5. Basili et al. [6] say that they will investigate how an ontological approach could support QA across a "federation" of sites within the same domain. In contrast, AquaLog will not assume that the ontologies refer to the same domain; they can have overlapping domains or may refer to different domains. But similarly to [6] AquaLog has to deal with the heterogeneity introduced by ontologies themselves: "since each node has its own version of the domain ontology, the task of passing a question from node to node may be reduced to a mapping task between (similar) conceptual representations".

The knowledge-based approach described in Ref. [12] is to "augment on-line text with a knowledge-based question-answering component, [. . .] allowing the system to infer answers to users' questions which are outside the scope of the prewritten text". It assumes that the knowledge is stored in a knowledge base and structured as an ontology of the domain. Like many of the systems we have seen it has a small collection of generic question types which it knows how to answer. Question types are associated with concepts in the KB. For a given concept, the question types which are applicable are those which are attached either to the concept itself or to any of its superclasses. A difference between this system and others we have seen is the importance it places on building a scenario, part assumed and part specified by the user via a dialogue in which the system prompts the user with forms and questions based on an answer schema that relates back to the question type. The scenario provides a context in which the system can answer the query. The user input is thus considerably greater than we would wish to have in AquaLog.

9.5. Conclusions of existing approaches

Since the development of the first ontological QA system LUNAR (a syntax-based system where the parsed question is directly mapped to a database expression by the use of rules [3]) there have been improvements in the availability of lexical knowledge bases, such as WordNet, and shallow, modular and robust NLP systems, like GATE. Furthermore, AquaLog is based on the vision of a Web populated by ontologically tagged documents. Many closed domain NL interfaces are very rich in NL understanding and can handle questions that are more complex than the ones handled by the current version of AquaLog. However, AquaLog has a very light and extendable NL interface that allows it to produce triples after only shallow parsing. The main difference between the systems is related to portability. Later NLDBI systems use intermediate representations therefore although the front end is portable (as Copestake [14] states "the grammar is, to a large extent, general purpose, it can be used for other domains and for other applications") the back end is dependent on the database, so normally longer configuration times are required. AquaLog, in contrast with closed domain systems, is completely portable. Moreover, the AquaLog disambiguation techniques are sufficiently general to be applied in different domains. In AquaLog, disambiguation is regarded as part of the translation process, if the ambiguity is not solved by domain knowledge, then the ambiguity is detected and the user is consulted before going ahead (to choose between alternative reading on terms, relations or modifier attachment).

AquaLog is complementary to open domain QA systems. Open QA systems use the Web as the source of knowledge

and provide answers in the form of selected paragraphs (where the answer actually lies) extracted from very large open-ended collections of unstructured text. The key limitation of an ontology-based system (as for closed-domain systems) is that it presumes the knowledge the system is using to answer the question is in a structured knowledge base in a limited domain. However, AquaLog exploits the power of ontologies as a model of knowledge and the availability of semantic markup offered by the SW to give precise, focused answers rather than retrieving possible documents or pre-written paragraphs of text. In particular, semantic markup facilitates queries where multiple pieces of information (that may come from different sources) need to be inferred and combined together. For instance, when we ask a query such as "what are the homepages of researchers who have an interest in the Semantic Web?", we get the precise answer. Behind the scenes, AquaLog is not only able to correctly understand the question but is also competent to disambiguate multiple matches of the term "researchers" on the SW and give back the correct answers by consulting the ontology and the available metadata. As Basili argues in Ref. [6] "open domain systems do not rely on specialized conceptual knowledge as they use a mixture of statistical techniques and shallow linguistic analysis. Ontological Question Answering Systems [. . .] propose to attack the problem by means of an internal unambiguous knowledge representation".

Both open-domain QA systems and AquaLog classify queries: in AquaLog based on the triple format, in open domain based on the expected answer (person, location) or hierarchies of question types based on types of answer sought (what, why, who, how, where. . .). The AquaLog classification includes not only information about the answer expected (a list of instances, an assertion, etc.) but also depending on the triples they generate (number of triples, explicit versus implicit relationships or query terms) and how they should be resolved. It groups different questions that can be presented by equivalent triples. For instance, the query "Who works in akt?" is the same as "Which are the researchers involved in the akt project?"

We believe that the main benefit of an ontology-based QA system on the SW, when compared to other kind of QA systems, is that it can use the domain knowledge provided by the ontology to cope with words apparently not found in the KB and to cope with ambiguity (mapping vocabulary or modifier attachment).

10. Conclusions

In this paper we have described the AquaLog question answering system, emphasizing its genesis in the context of semantic web research. The key ingredient to ontology-based QA systems, and to the most accurate non-ontological QA systems is the ability to capture the semantics of the question and use it in the extraction of answers in real time. AquaLog does not assume that the user has any prior information about the semantic resource. AquaLog's requirement of portability makes it impossible to have any pre-formulated assumptions about the ontological structure of the relevant information, and the problem cannot be addressed by the specification of static mapping rules. AquaLog presents an elegant solution in which different strategies are combined together to make sense of an NL query which respect to the universe of discourse covered by the ontology. It provides precise answers to complex queries, where multiple pieces of information need to be combined together at run time. AquaLog makes sense of query terms/relations, expressed in terms familiar to the user, even when they appear not to have any match. Moreover, the performance of the system improves over time in response to a particular community jargon.

Finally, its ontology portability capabilities make AquaLog a suitable NL front-end for a semantic intranet where a shared dynamic organizational ontology is used to describe resources. However, if we consider the Semantic Web in the large there is a need to compose information from multiple resources that are autonomously created and maintained. Our future directions on AquaLog and ontology-based QA are evolving from relying on a single ontology at a time to opening up to harvest the rich ontological knowledge available on the Web, allowing the system to benefit from and combine knowledge from the wide range of ontologies that exist on the Web.

Acknowledgements

This work was funded by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), the Designing Information Extraction for KM (Dot.Kom) project, and the Open Knowledge (OK) project. AKT is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. Dot.Kom was sponsored by the European Commission as part of the Information Society Technologies (IST) programme under grant number IST-2001034038. OK is sponsored by European Commission as part of the Information Society Technologies (IST) programme under grant number IST-2001-34038. The authors would like to thank Yuangui Lei for technical input and Marta Sabou for all her useful input and her valuable revision of this paper.

Appendix A. Examples of NL queries and equivalent triples

wh-generic term	Linguistic triple	Ontology triple[a]
Who are the researchers in the semantic web research area?	⟨*person/organization*, researchers, semantic web research area⟩	⟨*researcher*, has-research-interest, semantic-web-area⟩
Name the planet stories that are related to akt	⟨*planet stories*, related to, akt⟩	⟨*kmi-planet-news-item*, mentions-project, akt⟩
What are the phd students working for buddy space?	⟨*phd students*, working, buddy space⟩	⟨*phd-student*, has-project-member/has-project-leader, buddyspace⟩

Appendix A (*Continued*)

wh-unknown term	Linguistic triple	Ontology triple
Show me the job title of Peter Scott	⟨*which is*, job title, peter scott⟩	⟨*which is*, has-job-title, peter-scott⟩
What are the projects of Vanessa?	⟨*which is*, projects, vanessa⟩	⟨*project*, has-proj ect-member/has-proj ect-leader, vanessa⟩
wh-unknown relation	**Linguistic triple**	**Ontology triple**
Are there any projects about semantic web?	⟨*projects*, ?, semantic web⟩	⟨*project*, addresses-generic-area-of-interest, semantic-web-area⟩
Where is the Knowledge Media Institute?	⟨*location*, ?, knowledge media institute⟩	No relations found in the ontology
Description	**Linguistic triple**	**Ontology triple**
Who are the academics?	⟨*who/what is*, *?*, academics⟩	⟨*who/what is*, *?*, academic-staff-member⟩
What is an ontology?	⟨*who/what is*, ?, ontology⟩	⟨*who/what is*, *?*, ontologies⟩
Affirmative/negative	**Linguistic triple**	**Ontology triple**
Is Liliana a phd student in the dip project?	⟨liliana, phd student, dip project⟩	⟨liliana-cabral (phd-student) has-project-member, dip⟩
Has Martin Dzbor any interest in ontologies?	⟨martin dzbor, has any interest, ontologies⟩	⟨martin-dzbor, has-research-interest, ontologies⟩
wh-3 terms	**Linguistic triple**	**Ontology triple**
Does anybody work in semantic web on the dotkom project?	⟨*person/organization*, works, semantic web, dotkom project⟩	⟨*person*, has-research-interest, semantic-web-area⟩ ⟨*person*, has-proj ect-member/has-proj ect-leader, dot-kom⟩
tell me all the planet news written by researchers in akt	⟨*planet news*, written, researchers, akt⟩	⟨*kmi-planet-news-item*, has-author, researcher⟩ ⟨*researcher*, has-proj ect-member/has-proj ect-leader, akt⟩
wh-3 terms (first clause)	**Linguistic triple**	**Ontology triple**
Which projects on information extraction are sponsored by eprsc	⟨*projects*, sponsored, information extraction, epsrc⟩	⟨*project*, addresses-generic-area-of-interest, information-extraction⟩ ⟨*project*, involves-organization, epsrc⟩
wh-3 terms (unknown relation)	**Linguistic triple**	**Ontology triple**
Is there any publication about magpie in akt?	⟨*publication*, ?, magpie, akt⟩	*publication*, mentions-project/has-key-publication, has-publication, akt⟩ ⟨*publication*, mentions-project, akt⟩
wh-unknown term (with clause)	**Linguistic triple**	**Ontology triple**
What are the contact details from the KMi academics in compendium?	⟨*which is*, contact details, kmi academics, compendium⟩	⟨*which is*, has-web-address/has-email-address/has-telephone-number, kmi-academic-staff-member⟩ ⟨*kmi-academic-staff-member*, has-project-member, compendium⟩
wh-combination (and)	**Linguistic triple**	**Ontology triple**
does anyone has interest in ontologies and is a member of akt?	⟨*person/organization*, has interest, ontologies⟩ ⟨*person/organization*, member, akt⟩	⟨*person*, has-research-interest, ontologies⟩ ⟨*research-staff-member*, has-project-member, akt⟩
wh-combination (or)	**Linguistic triple**	**Ontology triple**
Which academics work in akt or in dotcom?	⟨*academics*, work, akt⟩ ⟨*academics*, work, dotcom⟩	⟨*academic-staff-member*, has-project-member, akt⟩ ⟨*academic-staff-member*, has-project-member, dot-kom⟩
wh-combination conditioned	**Linguistic triple**	**Ontology triple**
Which KMi academics work in the akt project sponsored by epsrc?	⟨*kmi academics*, work, akt project⟩ ⟨*which is*, sponsored, epsrc⟩	⟨*kmi-academic-staff-member*, has-project-member, akt⟩ ⟨*project*, involves-organization, epsrc⟩
Which KMi academics working in the akt project are sponsored by epsrc?	⟨*kmi academics*, working, akt project⟩ ⟨*kmi academics*, sponsored, epsrc⟩	⟨*kmi-academic-staff-member*, has-project-member, akt⟩ ⟨*kmi-academic-staff-member*, has-affiliated-person, epsrc⟩
Give me the publications from phd students working in hypermedia	⟨*which is*, publications, phd students⟩ ⟨*which is*, working, hypermedia⟩	*publication*, mentions-person/has-author, phd student⟩ ⟨*phd-student*, has-research-interest, hypermedia⟩
wh-generic with wh-clause	**Linguistic triple**	**Ontology triple**
What researchers, who work in compendium, have interest in hypermedia?	⟨*researchers*, work, compendium⟩ ⟨*researchers*, has-interest, hypermedia⟩	⟨*researcher*, has-project-member, compendium⟩ ⟨*researcher*, has-research-interest, hypermedia⟩

Appendix A (*Continued*)

2-patterns	Linguistic triple	Ontology triple
What is the homepage of Peter who has interest in semantic web?	⟨*which is*, homepage, peter⟩ ⟨*person/organization*, has interest, semantic web⟩	⟨*which is*, has-web-address, peter-scott⟩ ⟨*person*, has-research-interest, semantic-web-area⟩
Which are the projects of academics that are related to the semantic web?	⟨*which is*, projects, academics⟩ ⟨*which is*, related, semantic web⟩	⟨*project*, has-project-member, academic-staff-member⟩ ⟨*academic-staff-member*, has-research-interest, semantic-web-area⟩

[a] Using the KMi ontology.

References

[1] AKT Reference Ontology, http://kmi.open.ac.uk/projects/akt/ref-onto/index.html.

[2] I. Androutsopoulos, G.D. Ritchie, P. Thanisch, MASQUE/SQL—an efficient and portable natural language query interface for relational databases, in: P.W. Chung, G. Lovegrove, M. Ali (Eds.), Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, UK, Gordon and Breach Publishers, 1993, pp. 327–330.

[3] I. Androutsopoulos, G.D. Ritchie, P. Thanisch, Natural language interfaces to databases—an introduction, Nat. Lang. Eng. 1 (1) (1995) 29–81.

[4] AskJeeves, AskJeeves: http://www.ask.co.uk.

[5] G. Attardi, A. Cisternino, F. Formica, M. Simi, A. Tommasi, C. Zavattari, PIQASso: PIsa question answering system, in: Proceedings of the text Retrieval Conferemce (Trec-10), 599-607, NIST, Gaithersburg, MD, November 13–16, 2001.

[6] R. Basili, D.H. Hansen, P. Paggio, M.T. Pazienza, F.M. Zanzotto, Ontological resources and question data in answering, in: Workshop on Pragmatics of Question Answering, held jointly with NAACL, Boston, MA, May, 2004.

[7] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001).

[8] J. Burger, C., Cardie, V., Chaudhri, et al., Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A), NIST Technical Report, 2001. http://www.ai.mit.edu/people/jimmylin/%0Apapers/Burger00-Roadmap.pdf.

[9] R.D. Burke, K.J. Hammond, V. Kulyukin, Question answering from frequently-asked question files: experiences with the FAQ finder system, Tech. Rep. TR-97-05, Department of Computer Science, University of Chicago, 1997.

[10] J. Chu-Carroll, D. Ferrucci, J. Prager, C. Welty, Hybridization in question answering systems, in: M. Maybury (Ed.), New Directions in Question Answering, AAAI Press, 2003.

[12] P. Clark, J. Thompson, B. Porter, A knowledge-based approach to question-answering, in: In the AAAI Fall Symposium on Question-Answering Systems, CA, AAAI, 1999, pp. 43–51.

[13] W.W. Cohen, P. Ravikumar, S.E. Fienberg, A comparison of string distance metrics for name-matching tasks, in: IIWeb Workshop, 2003, http://www-2.cs.cmu.edu/~wcohen/postscript/ijcai-ws-2003.pdf.

[14] A. Copestake, K.S. Jones, Natural language interfaces to databases, Knowl. Eng. Rev. 5 (4) (1990) 225–249.

[15] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, GATE: a framework and graphical development environment for robust NLP tools and applications, in: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, 2002.

[16] De Boni, M. TREC 9 QA track overview.

[17] A.N De Roeck, C.J. Fox, B.G.T. Lowden, R. Turner, B. Walls, A natural language system based on formal semantics, in: Proceedings of the International Conference on Current Issues in Computational Linguistics, Pengang, Malaysia, 1991.

[18] Discourse Represenand then tation Theory. Jan Van Eijck, to appear in the 2nd edition of the Encyclopedia of Language and Linguistics, Elsevier, 2005.

[19] M. Dzbor, J. Domingue, E. Motta, Magpie—towards a semantic web browser, in: Proceedings of the 2nd International Semantic Web Conference (ISWC2003), Lecture Notes in Computer Science, 2870/2003, Springer-Verlag, 2003.

[20] EasyAsk: http://www.easyask.com.

[21] C. Fellbaum (Ed.), WordNet, An Electronic Lexical Database, Bradford Books, May, 1998.

[22] R. Guha, R. McCool, E. Miller, Semantic search, in: Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, 2003.

[23] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, P. Morarescu, Falcon—boosting knowledge for answer engines, in: Proceedings of the 9th Text Retrieval Conference (Trec-9), Gaithersburg, MD, November, 2000.

[24] L. Hirschman, R. Gaizauskas, Natural Language question answering: the view from here., Nat. Lang. Eng. 7 (4) (2001) 275–300 (special issue on Question Answering).

[25] E.H. Hovy, L. Gerber, U. Hermjakob, M. Junk, C.-Y. Lin, Question answering in Webclopedia, in: Proceedings of the TREC-9 Conference. NIST, Gaithersburg, MD, 2000.

[26] E. Hsu, D. McGuinness, Wine agent: semantic web testbed application, Workshop on Description Logics, 2003.

[27] A. Hunter, Natural language database interfaces, Knowl. Manage. (2000).

[28] H. Jung, G. Geunbae Lee, Multilingual question answering with high portability on relational databases, IEICE Trans. Inform. Syst. E86-D (2) (2003) 306–315.

[29] JWNL (Java WordNet library) http://sourceforge.net/projects/jwordnet.

[30] J. Kaplan, Designing a portable natural language database query system, ACM Trans. Database Syst. 9 (1) (1984) 1–19.

[31] B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A.J. McFarland, B. Temelkuran, Omnibase: uniform access to heterogeneous data for question answering, in: Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB), 2002.

[32] B. Katz, J. Lin, REXTOR: a system for generating relations from natural language, in: Proceedings of the ACL-2000. Workshop of Natural Language Processing and Information Retrieval (NLP(IR)), 2000.

[33] B. Katz, J. Lin, Selectively using relations to improve precision in question answering, in: Proceedings of the EACL-2003. Workshop on Natural Language Processing for Question Answering, 2003.

[34] D. Klein, C.D. Manning, Fast Exact Inference with a Factored Model for Natural Language Parsing, Adv. Neural Inform. Process. Syst. 15 (2002).

[35] Y. Lei, M. Sabou, V. Lopez, J. Zhu, V. Uren, E. Motta, An infrastructure for acquiring high quality semantic metadata, in: Proceedings of the 3rd European Semantic Web Conference, Montenegro, 2006.

[36] K.C. Litkowski, Syntactic clues and lexical resources in question-answering, in: E.M. Voorhees, D.K. Harman (Eds.), Information Technology: The Ninth TextREtrieval Conferenence (TREC-9), NIST Special Publication 500-249, National Institute of Standards and Technology, Gaithersburg, MD, 2001, pp. 157–166.

[37] V. Lopez, E. Motta, V. Uren, PowerAqua: fishing the semantic web, in: Proceedings of the 3rd European Semantic Web Conference, MonteNegro, 2006.

[38] V. Lopez, E. Motta, Ontology driven question answering in AquaLog, in: Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems, Manchester, England, 2004.

[39] P. Martin, D.E. Appelt, B.J. Grosz, F.C.N. Pereira, TEAM: an experimental transportable natural–language interface, IEEE Database Eng. Bull. 8 (3) (1985) 10–22.

[40] D. Mc Guinness, F. van Harmelen, OWL Web Ontology Language Overview. W3C Recommendation, 10, 2004 http://www.w3.org/TR/owl-features/.

[41] D. Mc Guinness, Question answering on the semantic web, IEEE Intell. Syst. 19 (1) (2004).

[42] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
[43] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, V. Rus, LASSO: a tool for surfing the answer net, in: Proceedings of the Text Retrieval Conference (TREC-8), November, 1999.
[45] M. Pasca, S. Harabagiu, The informative role of Wordnet in open-domain question answering, in: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL), 2001.
[46] A.M. Popescu, O. Etzioni, H.A. Kautz, Towards a theory of natural language interfaces to databases, in: Proceedings of the 2003 International Conference on Intelligent User Interfaces, Miami, FL, USA, January 12–15, 2003, pp. 149–157.
[47] RDF: http://www.w3.org/RDF/.
[48] K. Srihari, W. Li, X. Li, Information extraction supported question-answering, in: T. Strzalkowski, S. Harabagiu (Eds.), Advances in Open-Domain Question Answering, Kluwer Academic Publishers, 2004.
[49] V. Tablan, D. Maynard, K. Bontcheva, GATE—A Concise User Guide. University of Sheffield, UK. http://gate.ac.uk/.
[50] W3C, OWL Web Ontology Language Guide: http://www.w3.org/TR/2003/CR-owl-guide-0030818/.
[51] R. Waldinger, D.E. Appelt, et al., Deductive question answering from multiple resources, in: M. Maybury (Ed.), New Directions in Question Answering, AAAI Press, 2003.
[52] WebOnto project: http://plainmoor.open.ac.uk/webonto.
[53] M. Wu, X. Zheng, M. Duan, T. Liu, T. Strzalkowski, Question answering by pattern matching, web-proofing, semantic form proofing. NIST Special Publication, in: The 12th Text Retrieval Conference (TREC), 2003, pp. 255–500.
[54] Z. Zheng, The answer bus question answering system, in: Proceedings of the Human Language Technology Conference (HLT2002), San Diego, CA, March 24–27, 2002.