AquaLog: An Ontology-Portable Question Answering System for the Semantic Web

Vanessa Lopez, Michele Pasin, and Enrico Motta

Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes,
MK7 6AA, United Kingdom
{v.lopez, m.pasin, e.motta}@open.ac.uk

Abstract. As semantic markup becomes ubiquitous, it will become important 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. AquaLog is a portable question-answering system which takes queries expressed in natural language and an ontology as input and returns answers drawn from the available semantic markup. We say that AquaLog is portable, because the configuration time required to customize the system for a particular ontology is negligible. AquaLog combines several powerful techniques in a novel way to make sense of NL queries and to map them to semantic markup. Moreover it also includes a learning component, which ensures that the performance of the system improves over time, in response to the particular community jargon used by the end users. In this paper we describe the current version of the system, in particular discussing its portability, its reasoning capabilities, and its learning mechanism.

1 Introduction

The semantic web vision [1] is one in which rich, ontology-based semantic markup is widely available, thus opening the way to novel, sophisticated forms of question answering. However, much work on ontology-driven QA tends to focus on the use of ontologies to support query expansion in information retrieval [2], rather than on exploiting the availability of semantic statements to provide precise answers to complex queries. In particular, a knowledge based QA system can help with answering questions requiring situation-specific knowledge, where multiple pieces of information need to be inferred and combined at run time, rather than simply having a pre-written paragraph of text retrieved [3].

AquaLog is a portable question-answering system which takes queries expressed in natural language and an ontology as input and returns answers drawn from the available ontology-compliant semantic markup. We say that AquaLog is portable, because the configuration time required to customize the system for a particular ontology is negligible. AquaLog combines several powerful techniques in a novel way to make sense of NL queries and to map them to semantic markup. Specifically, it makes use of the GATE NLP platform, string metrics algorithms [4], WordNet

A. Gómez-Pérez and J. Euzenat (Eds.): ESWC 2005, LNCS 3532, pp. 546 – 562, 2005.

[5, 6], 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, which ensures that the performance of the system improves over time, in response to the particular community jargon used by the end users.

AquaLog is implemented in Java as a web application, using a client-server architecture. Moreover, it provides an API, which allows future integration in other platforms and independent use of its components. A key feature of AquaLog is the use of a plug-in mechanism, which allows AquaLog to be configured for different KR languages.

In this paper we describe the current version of the system, in particular discussing its portability, its reasoning capabilities, and its learning mechanism.

The paper is organized as follows: section 2 describes the AquaLog architecture. Section 3 describes the Linguistic Component embedded in AquaLog. Section 4 describes the novel Relation Similarity Service and Learning Mechanism. Section 5 describes a case of integration with Web Services. Section 6 describes the evaluation scenario, followed by discussion and directions for future work. Section 7 describes related work. Finally, section 8 re-iterates the main contributions of this work.

2 The Architecture

At a coarse-grained level of abstraction, the AquaLog architecture can be characterized as a waterfall model, during which a NL query gets translated into a set of intermediate, triple-based representations, query-triples, and then these are translated into ontology-compatible triples, as shown in figure 1. There are two main reasons for adopting a triple-based data model: first of all, it is possible to represent most queries as triples. Secondly, RDF-based knowledge representation (KR) formalisms for the semantic web, such as RDF itself [7] or OWL [8] 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 this data model. However AquaLog triples also have additional features in order to facilitate the reasoning about the answer, such as the voice and tense of the relation and the category. 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[1]?" 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 a quick overview of the two main processing modules in AquaLog: the *linguistic component* and the *relation similarity service*. To illustrate

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

the system we will consider as test case the semantic web site currently under construction at the knowledge media institute, see http://plainmoor.open.ac.uk: 8080/ksw, which relies on an ontology which characterizes the key aspects of academic life. Specifically the ontology includes classes and relations to describe projects, technologies, people, news, events, organizations, publications, and research areas. The full specification of the ontology can be found at http://plainmoor. open.ac.uk: 8080/ ksw/ontologies.html. The semantic markup is generated automatically by mining text resources and representing the information held in departmental databases, in terms of the ontology.

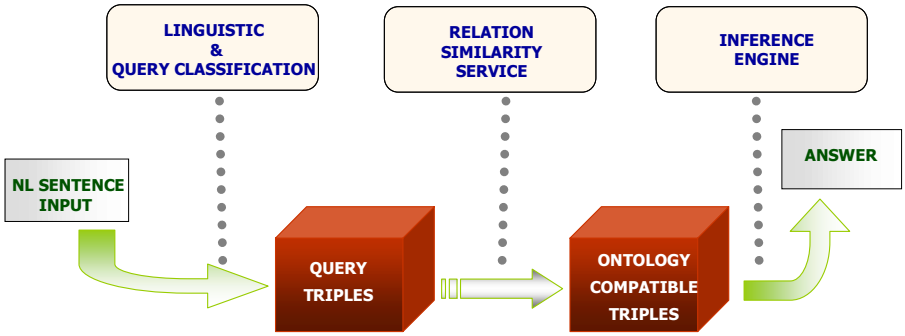

Fig. 1. The AquaLog Data Model

3 Linguistic Component

The Linguistic Component task is to map the NL input query to the Query-Triple. AquaLog uses the GATE [9, 10] infrastructure and resources in order to parse the question 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. These annotations include information about sentences, tokens, nouns and verbs. When developing AquaLog we extended the set of annotations returned by GATE, by identifying terms, relations, question indicators (which/who/when, etc.) and patterns or types of questions. This is achieved through the use of *JAPE grammars*, 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.

Thanks to this architecture it is possible to extend the NL capability of the system in a relatively easy way (NL scalability). Currently, the Linguistic Component, through the *JAPE grammars*, dynamically identifies 23 different linguistic categories or intermediate representations, 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 not information about the type of the expected answer is provided; etc.

In some cases, e.g. When interpreting the query "list all the projects in KMi about Semantic Web", the linguistic components cannot resolve the ambiguity associated with the NL query (it cannot identify the constituent to which each modifier has to be attached) and therefore it simply passes the ambiguity on to the Relation Similarity Service (RSS), which can use the ontology or ask the user to solve the ambiguity.

It is important to emphasize that, at this stage 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, which we use mainly because at this stage we do not have to worry about getting the representation right in respect to the specific domain knowledge. The role of the intermediate representation is simply to provide an easy way to manipulate input for the RSS. This design choice ensures the easy portability of the system with respect to both ontologies and natural languages.

4 Relation Similarity Service

This 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. Essentially the RSS tries to make sense of the input query by looking at the structure of the ontology and the information available on the semantic web, 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, as explained in a later section.

An important aspect of the RSS is that it is interactive. In other words, when the RSS is not sure about how to disambiguate between two or more possible terms or relations in order to interpret a query it will ask the user for disambiguation.

Relations and concepts' names are identified and mapped within the ontology through the RSS and the Class Similarity Service (CSS) respectively. The latter is a sub-module of the RSS, which deals with mapping linguistic terms to classes. Proper names, instead, are mapped into instances by means of the use of string distance metrics algorithms [4]. If this mapping fails a partial solution is implemented for affirmative/negative type of questions, where we make sense of questions in which only one of two instances is recognized. For instance, in the query "is Enrico working in ibm?", "Enrico" could be mapped into "enrico-motta" in the KB but "ibm" is not found. The answer will output 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

[11]. Whether 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 it. The system scans the answers for words denoting instances which are represented in the knowledge base, and then adds hyperlinks to these words/phrases, indicating that the user can click on them. In fact, the RSS is designed to provide justifications for every step of the user interaction. This is crucial to ensure user acceptance of the system.

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 KMi?" is parsed into <person/organization, secretary, kmi>, following purely linguistic criteria. Then, the first step for the RSS is to identify, in the target KB that "kmi" is actually a "research-institute" called "knowledge-media-institute". Once a successful match is found, the problem becomes to find a relation which links the class research institute (or its superclass organization) to class person (or any of its subclasses, such as academic, student, etc...) or to class organization, by analyzing the taxonomy and relationships in the target KB. However, in this particular case there is a successful matching in the KB for *secretary*, even if secretary is not a relation but a subclass of person. The RSS reasons about the mismatch, re-classifies the intermediate query and generates the correct logical query, in compliance which the ontology, which is organized in terms of <secretary, works-for, kmi>.

Whenever multiple relations are possible candidates for interpreting the query, if the ontology does not provide ways to further discriminate between them, string matching is used to determine the most likely candidate, using the relation name, the learning mechanism, or eventual aliases provided by lexical resources such as WordNet [12]. If no relations are found by using these methods, then the user is asked to choose from the current list of candidates.

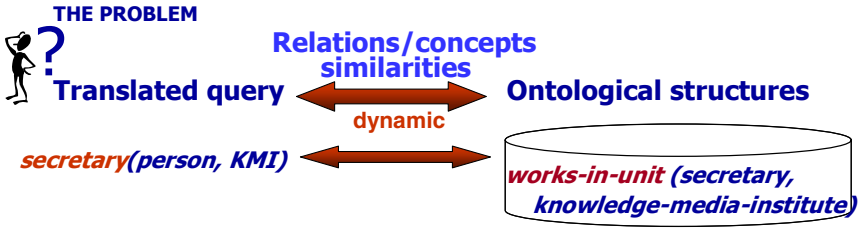

Fig. 2. Scheme for mapping a Query-Triple into an Onto-Triple

Another case is the one in which a query map to a set of triples. In these cases the ambiguity can also be related to the way the triples are linked. The RSS deals with

these cases both by analyzing the structure in the ontology and through the use of heuristics.

For example, let's consider the query "which news stories have been written by researchers in akt?". To handle this case the RSS uses a heuristic which suggest the modifier "in akt" to be attached to the closest term that is represented by a class or non-ground term in the ontology, in this case the class "researchers".

An example of query disambiguation using a combination of linguistic and semantic information from the ontology can be seen in Figure 3. Here a user has asked "Who is the researcher in akt who is interested in the Semantic Web?". This query is syntactically ambiguous, because the second clause, "who is interested in the Semantic Web", could syntactically link to either the researcher or "akt". Because AquaLog knows that "who" can only be a person or an organization, it correctly links it to "researcher", rather than "akt". However, there can be other situations where the disambiguation cannot be resolved by using the use of linguistic and/or heuristics and/or the context or semantics in the ontology, as for example in the query "which academic works with peter who has an interest in the semantic web?". In this case since "academic" and "peter" are respectively a subclass and an instance of "person", the sentence is truly ambiguous. In fact, it can be understood either as a combination of the resulting lists of the two questions "which academic works with peter" and "which academic has an interest in the 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's feedback is always required.

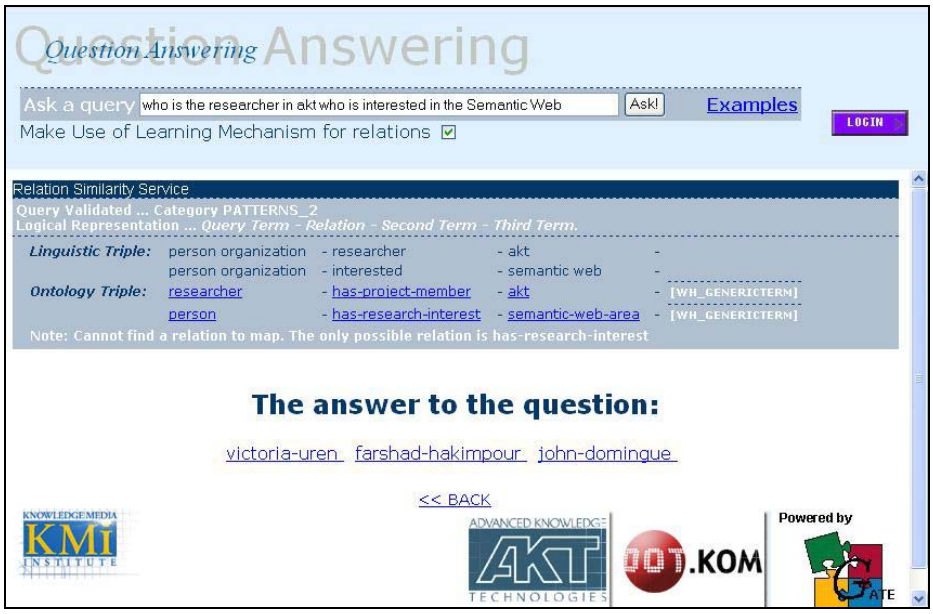

Fig. 3. Example of context disambiguation by the RSS

4.1 Class Similarity Service

The use of string metrics to map the generic term of the linguistic triple into a term in the ontology may not be enough. Therefore, an additional combination of methods to get synonyms (such as WordNet or our own lexicon) may be used in order to obtain the possible candidates in the ontology. This 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 a later section). The only requirement to execute this learning mechanism for classes is the availability of the ontology mapping for one of the two terms of the triple. In this way, through the ontology relationships that are valid for this term, we can identify a set of possible candidate terms that can complete the triple. User's feedback is required to select whether one of the candidate terms is the one we are looking for, so that the system is able to learn it for future occasions.

4.2 Learning Mechanism

Since the universe of discourse we are working with is determined by and limited to the particular ontology used, there will normally 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 a user-specific "jargon" found in the linguistic triple. In such a case, it is 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 very common and highly generic example, in our departmental ontology, is the relation *works-for*, to which users normally relate a number of different expressions: *is working, works, collaborate, is involved*. In all these cases the user is asked to disambiguate the relation (choosing from the set of ontology relations consistent with the two question's arguments) and decide if a new mapping should be learned between his/her natural-language-universe and the ontology-language-universe.

4.2.1 Architecture

The learning mechanism in AquaLog consists of two different methods, the *learning* and the *matching* (fig. 4). 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 it 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 particular representation of the

context and defines a structured space of hypothesis analogue to that one of a version space[2] [13].

In future work, this context will be further extended to provide more granularity and semantic expressiveness.

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 thus 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 basically just 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 in 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 it relies on the RSS classification.

4.2.2 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 meaning different things.

For example, let's consider the question "Who collaborates with the knowledge media institute?" and assume that the system is not able to solve the linguistic ambiguity of the word "collaborate". The first time, some help from the user is needed, 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 now a professor, who asks the system the same question "Who collaborates with the knowledge media institute?", but is referring to other research

[2] A version space is an inductive learning technique proposed by Mitchell in order to represent the consistency of a set of hypothesis with a target concept.

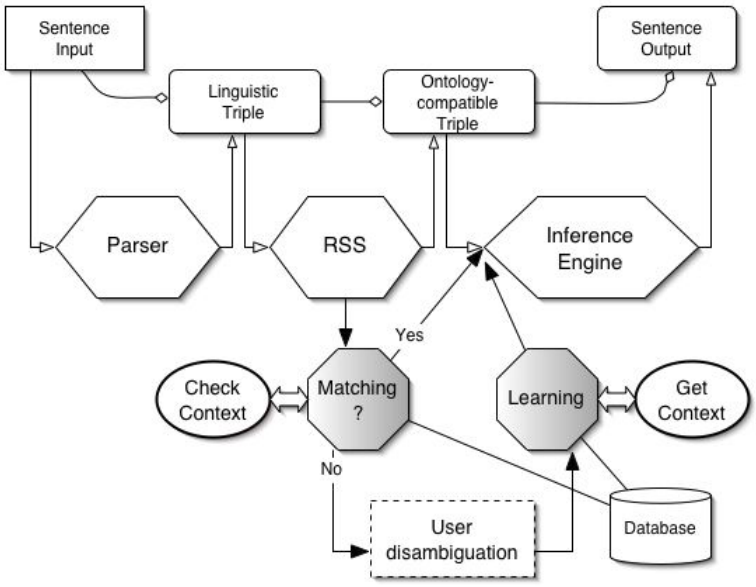

Fig. 4. The learning mechanism architecture

labs or academic units involved with the knowledge media institute. In fact, when asked to choose from the list of possible ontology relations, he/she will possibly enter "works-in-the-same-project".

The problem, so, is to maintain the two mappings separated while still providing some kind of generalization. This is achieved through the definition of the question's 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 are classes or 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 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.

4.2.3 Context Generalization

Of course, 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 also in this case.

In order to generalize these results the strategy adopted is to record the most generic classes in the ontology which corresponds to the two triple's arguments, and, at the same time, can handle the same relation. Namely, 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, since only the highest classes of an ontology's branch are kept, all the questions similar to the ones we have seen will fall within the same set, because 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 the two interpretations <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.

4.2.4 User 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. For example, during a specific user-session it is possible to delete some previous recorded mappings, action that is normally not permitted to the generic user. This latter has in fact the roughest access to the learned material: having no constraints on the user field, the database query will return many more mappings and, quite likely, also meanings that are not desired.

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

5 Integration with Web Services

As we said before, every item in the onto-triple is an entry point to the knowledge base or to the ontology. Therefore, items are clickable and the user can get more information about them. Optionally, AquaLog can be configured to use Semantic Web Services in order to get more information about a particular item (i.e. instance or concept), when required. Here AquaLog uses the same mechanism used by Magpie [14], accessing services published against the same ontology and KB.

6 Evaluation Scenario

AquaLog allows a user who has a question in mind and knows 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 vocabularies, or to know the structure of the knowledge base. However, as pointed in [11], although they have to have some idea of the contents of the domain they may have some misconceptions. Therefore some process of familiarization is normally required.

A full evaluation of AquaLog requires both an evaluation of its query answering ability as well an evaluation of the overall user experience. Moreover, because one of our key aims is to make AquaLog an interface for the semantic web, the portability across ontologies will also have to be evaluated formally.

For the first version of AquaLog [15] we performed an initial study, whose aim was to assess to what extent the AquaLog application built using AquaLog with the AKT ontology and the KMi knowledge base satisfied user expectations about the range of questions the system should be able to answer. 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 the system but also 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 the system with respect to user needs, we did not give them much information about the linguistic ability of the system.

We collected in total 76 different questions, 37 of which were handled correctly by AquaLog, i.e., 48.68% of the total. This was a pretty good result, considering that no linguistic restrictions were imposed on the questions.

As pointed in [27] it is very difficult to devise a sublanguage which is sufficiently expressive, yet avoids ambiguity and seems reasonable natural. Furthermore the limitations on linguistic coverage will not be obvious for the user and as a result, independently of whether a particular set of queries is answered or not, the system becomes unusable. Therefore, the conclusion of this previous study was that it was

absolutely crucial to improve the linguistic coverage of the system, which accounted for 69% of the failures.

For the current version of AquaLog, the linguistic coverage (and therefore data model and similarity services) has been extended considerably. At the same time AquaLog can now also deal with the ambiguity problems, derived from the use of more extensive grammars.

However, in this previous study we also identified failures due to a lack of services defined over ontologies (accounted for 20.5% of the errors). 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. 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.

No work has been done yet in relation to the service failures, which remains a future line of work for future versions of the system.

In order to evaluate the portability of the system we interfaced AquaLog to the Wine Ontology [16], an ontology used to illustrate the specification of the OWL W3C recommendation. The experiment confirmed the thesis that AquaLog is ontology independent, 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, which must be addressed in the near future. For instance, a direct question like "which wines are recommended with cakes" will fail because there is not a direct relation between wines and desserts, as there is a mediating concept called "mealcourse". 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 does not have much information instantiated, and as a result no answer can be found for most of the questions. However, it is a good test case for the Linguistic and Similarity Components responsible for creating the ontology compliance triple (from which an answer can be inferred in a relatively easy way).

7 Related Work

7.1 Close-Domain Natural Language Interfaces

This scenario is of course very similar to asking natural language queries to databases (NLDB), which has long been an area of research in the artificial intelligence and database communities [17, 18, 19, 20, 21], even if as [22, 23] say "in the past decade has somewhat gone out of fashion". The use of natural language to access relational databases can be traced back from the late sixties and early seventies. In [22] a detailed overview of the state of the art for these systems can be found. The main difference between AquaLog and the latest generation of NLDB systems [24] is that AquaLog uses an intermediate representation throughout the entire process, from the representation of the user's query (NL front end) to the representation of an ontology compliant triple (through the use of similarity services), from which an answer can be

directly inferred. It takes advantage of the structure of ontologies in a way that makes the entire process highly portable.

PRECISE [25] 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 pairs and a relation token corresponds to either an attribute token or a value token. 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 contrast with PRECISE, AquaLog employs similarity services to interpret the user 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.

7.2 Open-Domain QA Systems

Most current work on question answering is somewhat different in nature from AquaLog as it concerns open-domain systems. However, there are linguistic problems common in most kinds of natural language understanding systems.

Most text based QA applications typically involve two steps [26]: 1. Identifying the semantic type of the entity sought by the question (a date, a person and so on); 2. Determining additional constraints on the answer entity, i.e. identifying key words or syntactic or semantic relations to be used in matching candidate answers. Various systems have, therefore built hierarchies of question types based on the types of answers sought [27, 28, 29, 30].

As pointed by R. Srihari et al. in [28]: (i) IE can provide solid support for QA; (ii) low-level IE like Named Entity (NE) tagging is often a necessary component (an analysis showed that over 80% out of 200 questions asked for an NE as a response); (iii) a robust natural language shallow parser provides a structural basis for handling questions; (iv) high-level domain independent IE, i.e., extraction of multiple relationships between entities, is expected to bring about a breakthrough in QA.

AquaLog also subscribes to point (iii), however the main two differences with open-domain systems are: (1) it is not necessary to build hierarchies or heuristics to recognize name 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 [28] 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 queries key words” are used.

Both AquaLog and open-domain systems attempt to find synonyms plus their morphological variants to the terms or key words. Also in both cases, at times, the rules leave ambiguity unresolved and produce non-deterministic output for the focus of the question or 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 beforehand. The main difference is that AquaLog classifies the question based on the kind of triple needed, while most of the open-domain QA systems classify questions according to their answer target [30] (person, location, date, ..). The triple contains information not only about the answer expected or focus, which is what we call the generic term 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 may belong to the same triple category. An efficient system should therefore group together equivalent questions types.

The best result of the TREC9 [31] were obtained by the system FALCON described in Harabaigiu et al. [32]. 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. The example shown in [32] identifies the expected answer type of the question "what do penguins eat?" to be food since it is the most widely used concept in the glosses of the subhierarchy of the noun *synset* {eating, feeding}. 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.

7.3 Open-Domain QA Systems Using Triple Representation

The START [33] system goal is also to extract answers from text. AquaLog 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 [33]: "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 et al. [34], called DIMAP, extracts "semantic relation triples" from a document. The semantic relation triple described consists of a discourse entity, a semantic relation that characterizes the entity's role in the sentence and a governing word (generally the word in the sentence that the discourse entity stood in relation to). The semantic relation and the governing words were not identified for all discourse entities, but a record for each entity was still added to the database sentence (on average 9.8 triples per sentence). The same analysis is performed to create a set of records for each question (in average 3.3 triples per sentence), in which one of the semantic relation triples contained an unbound variable as a discourse entity, corresponding to the type of question. DIMAP-QA converts the document into triples and AquaLog uses the ontology, which it 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 questions can be translated into one or two triples.

7.4 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 R. Basili [35] the possibility of building an ontology-based question answering system in the context of the semantic web is discussed. Open domain QA systems do not rely on specialized conceptual knowledge as they use a mixture of statistical techniques and shallow linguistic analysis. Ontological QA systems propose to attack the problem by means of an internal unambiguous knowledge representation. 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. The approach and scenario has many similarities with AquaLog. However, AquaLog is implemented on-line and has a 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.

8 Conclusion

In this paper we have described the AquaLog ontology-driven query answering system in the context of the Semantic Web scenario. AquaLog presents an elegant solution in which different strategies are combined together to make sense of an NL query with respect to the universe of discourse covered by the ontology. Its ontology portability capabilities make AquaLog a suitable NL front-end for the Semantic Web.

Acknowledgements

This work was partially supported by the Advanced Knowledge Technologies (AKT), which is sponsored by the UK Engineering and Physical Sciences Research Council and by the Dot.Kom project under grant IST-2001-34038. The authors would like to thank Yuangui Lei, Anne de Roeck, Davide Guidi, Dnyanesh Rajpathak, Martin Dzbor, John Domingue, Victoria Uren and Kalina Bontcheva for useful AquaLog related input and those members of the lab who took part in the evaluation.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American, 284 (5)* (2001) 33-43
2. Mc Guinness, D.: Question Answering on the Semantic Web. *IEEE Intelligent Systems, 19 (1)* (2004) 82-85
3. Clark, P., Thompson, J., Porter., B.: A Knowledge-Based Approach to Question-Answering. In the *AAAI Fall Symposium on Question-Answering Systems*, CA: AAAI. (1999) 43-51
4. Cohen, W., W., Ravikumar, P., Fienberg, S., E.: 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
5. Pasca, M., Harabagiu, S.: 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)
6. JWNL (Java WordNet library) http://sourceforge.net/projects/jwordnet
7. RDF: http://www.w3.org/RDF/
8. Mc Guinness, D., van Harmelen, F.: OWL Web Ontology Language Overview. W3C Recommendation 10 (2004) http://www.w3.org/TR/owl-features/
9. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: 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)
10. Tablan, V., Maynard, D., Bontcheva, K.: *GATE - A Concise User Guide.* University of Sheffield, UK. http://gate.ac.uk/
11. Copestake, A., Jones, K., S.: Natural language interfaces to databases. *Knowledge Engineering Review*, 5 (4) (1990) 225-249
12. Fellbaum, C. (Ed.), *WordNet, An Electronic Lexical Database.* Bradford Books, May, (1998)
13. Mitchell, T. M.: *Machine learning.* McGraw-Hill, New York (1997)
14. Dzbor, M., Domingue, J., Motta, E.: 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)
15. Lopez, V., Motta, E.: Ontology Driven Question Answering in AquaLog. *In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems*, Manchester, England (2004)
16. W3C, OWL Web Ontology Language Guide: http://www.w3.org/TR/2003/CR-owl-guide-0030818/
17. Burger, J., Cardie, C., Chaudhri, V., 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
18. Kaplan, J.: Designing a portable natural language database query system. *ACM Transactions on Database Systems*, 9 (1) (1984) 1-19
19. Androutsopoulos, I., Ritchie, G.D., and Thanisch, P.: MASQUE/SQL - An Efficient and Portable Natural Language Query Interface for Relational Databases. In Chung, P.W. Lovegrove, G. and Ali, M. (Eds.), *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems,* Edinburgh, U.K., Gordon and Breach Publishers (1993) 327-330

20. Chu-Carroll, J., Ferrucci, D., Prager, J., Welty, C.: Hybridization in Question Answering Systems. In Maybury, M. (Ed.), *New Directions in Question Answering*, AAAI Press, (2003)
21. Jung, H., Geunbae Lee, G.: Multilingual Question Answering with High Portability on Relational Databases. *IEICE transactions on information and systems,* E86-D (2) (2003) 306-315
22. Androutsopoulos, I., Ritchie, G.D., Thanisch P.: Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering,* 1 (1) (1995) 29-81
23. Hunter, A.: Natural language database interfaces. *Knowledge Management,* (2000)
24. De Roeck, A., N., Fox, C., J., Lowden, B., G., T., Turner, R., Walls, B.: A Natural Language System Based on Formal Semantics. *In Proceedings of the International Conference on Current Issues in Computational Linguistics*, Pengang, Malaysia, (1991)
25. Popescu, A., M., Etzioni, O., Kautz, H., A.: Towards a theory of natural language interfaces to databases. *In Proceedings of the International Conference on Intelligent User Interfaces*, Miami, FL, USA, Jan. 12-15 (2003) 149-157
26. Hirschman, L., Gaizauskas, R.: Natural Language question answering: the view from here. *Natural Language Engineering, Special Issue on Question Answering*, 7 (4) (2001) 275-300
27. Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., Rus, V.: LASSO: A Tool for Surfing the Answer Net, *in Proceedings of the Text Retrieval Conference (TREC-8)*, Nov. (1999)
28. Srihari, K., Li, W., Li, X.: Information Extraction Supported Question- Answering, In T. Strzalkowski & S. Harabagiu (Eds.), in *Advances in Open- Domain Question Answering*. Kluwer Academic Publishers (2004)
29. Hovy, E.H., Gerber, L., Hermjakob, U., Junk, M., Lin, C.-Y.: Question Answering in Webclopedia. *In Proceedings of the TREC-9 Conference.* NIST, Gaithersburg, MD (2000)
30. Wu, M., Zheng, X., Duan, M., Liu, T., Strzalkowski, T.: Question Answering by Pattern Matching, Web-Proofing, Semantic Form Proofing. NIST Special Publication: *The Twelfth Text REtrieval Conference (TREC*) (2003) 500-255
31. De Boni, M.: TREC 9 QA track overview.
32. Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., Morarescu, P.: Falcon - Boosting Knowledge for Answer Engines. *In Proceedings of the 9th Text Retrieval Conference (Trec-9)*, Gaithersburg, Maryland, Nov. (2000)
33. Katz, B., Felshin, S., Yuret, D., Ibrahim, A., Lin, J., Marton, G., McFarland A. J., Temelkuran, B.: 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)
34. Litkowski, K. C. Syntactic Clues and Lexical Resources in Question-Answering. In Voorhees, E. M. and Harman, D. K. (Eds) *Information Technology: The Ninth Text REtrieval Conferenence (TREC-9)*, NIST Special Publication 500-249. Gaithersburg, MD: National Institute of Standards and Technology (2001) 157-66
35. Basili, R., Hansen, D., H., Paggio, P., Pazienza M., T., Zanzotto F., M. Ontological resources and question answering *Workshop on Pragmatics of Question Answering, held jointly with NAACL 2004* Boston, Massachusetts, May (2004)