KNOWLEDGE MEDIA INSTITUTE

AQUA: A Knowledge-Based Architecture for a Question Answering System

Tech Report kmi-04-15

Maria Vargas-Vera and Enrico Motta

The Open University

AQUA: A Knowledge-Based Architecture for a Question Answering System

Maria Vargas-Vera and Enrico Motta

Knowledge Media Institute (KMi), The Open University,
Walton Hall, Milton Keynes MK7 6AA, England
{m.vargas-vera}@open.ac.uk
http://kmi.open.ac.uk/

Abstract

This paper describes AQUA, a question answering system. AQUA combines Natural Language processing (NLP), Ontologies, Logic, and Information Retrieval technologies in a uniform framework. AQUA makes intensive use of an ontology (which encode knowledge) in several parts of the question answering system. The ontology is used in the refinement of the initial query, the reasoning process (a generalization/specialization process using classes and subclasses from the ontology), and in the novel similarity algorithm. The similarity algorithm is a key feature of AQUA. It is used to find similarities between relations/concepts in the translated query and relations/concepts in the ontological structures. The similarities detected then allow the interchange of concepts or relations in a logic formula corresponding to the user query. In this way, we make the mapping between user's queries and ontological spaces. The AQUA architecture is flexible enough to allow that AQUA can be used as closed-domain and open domain question answering system.

1. Introduction

In recent years, the rise in popularity of the web has created a demand for services which help users to skip over all irrelevant information quickly. One of the services is question answering (QA), the technique of providing answers to specific questions. Given a question such as 'which country had the highest inflation rate in 2002?' a keyword-based search engines such as Google might present the user with web pages from the Financial Times, whereas a QA system would attempt to directly answer the question with the name of a country. On the web, a typical example of a QA system is Jeeves[1] (Askjeeves 2000) which allows users to ask questions in natural language. It looks up the user's question in its own database and returns the list of matching questions which it knows how to answer. The user then selects the most appropriate entry in the list. However, users would usually prefer a precise answer to a precise question. Therefore, a reasonable aim for an automatic system is to provide textual answers instead of a set of documents. In this paper we present AQUA a question answering system which amalgamates Natural Language Processing (NLP), Logic, Ontologies and Information Retrieval techniques to provide answers to queries in a specific domain in real time.

The first instantiation of our ontology-driven Question Answering System, AQUA (as QA-closed domain), is designed to answer questions about academic people and organizations. However, an important future target application of AQUA would be to answer questions posed within company intra-nets; for example, giving AQUA an ontology of computer systems might allow it to be used for trouble-shooting or configuration of computer systems.
AQUA is also designed to play an important role in the Semantic Web[2]. One of the goals of the Semantic Web is the ability to annotate web resources with semantic content. These annotations can then be used by a reasoning system to provide intelligent services to users. AQUA would be able to perform incremental

[1] http://www.ask.com/
[2] The goal of the Semantic Web is to help users or software agents to organize, locate and process content on the WWW.

markup of home pages with semantic content. These annotations can be written in RDF (Lassila et al. 1999, Hayes 2002) or RDFS (Brickley et al. 2000), notations which provide a basic framework for expressing meta-data on the web. We envision that AQUA can perform the markup concurrently with looking for answers, that is, AQUA can annotated pages as it finds them. In this way then, semantically annotated web pages can be cached to reduce search and processing costs.

The main contribution of AQUA is the intensive use of an ontology in several parts of the question answering system. The ontology is used

- in the refinement of the initial query
- in the reasoning process (a generalization/specialization process using classes and subclasses from the ontology), and
- in the (novel) similarity algorithm.

The last of these, the similarity algorithm, is a key feature of AQUA. It is used to find similarities between relations/concepts in the translated query and relations/concepts in the ontological structures. The similarities detected then allow the interchange of concepts or relations in the formulae. The ontology is used to provide an intelligent reformulation of the question, with the intent to reduce the chances of failure to answer the question.

In this paper, we describe the overall workings of AQUA. Briefly, queries formulated in plain English are translated by AQUA into logic formulae using the grammatical components obtained by our parser. AQUA then looks for an answer in different resources such as databases, a populated ontology (or knowledge base) and the Web. Currently, AQUA makes use of an inference engine which is based on the Resolution algorithm. AQUA uses also our similarity algorithm which is based on both the ontological structures and instances of the ontology, a WordNet thesaurus and the Dice coefficient.
Finally, AQUA also has facilities for analyzing and explaining proofs. The explanation is provided both in pseudo-natural language and as a visualization. Throughout this paper, we will illustrate AQUA using queries based on the AKT reference ontology[3].

Our current work is focusing on embedding AQUA in the AKT semantic portal which will offer a variety of services to support academics in knowledge intensive tasks. A semantic portal can be seen as an entry point to knowledge resources that may be distributed across several locations. Such portal should provide means for navigating through all these resources in an easy way, since it may be used by users with all levels of computing knowledge. This semantic portal will offer users services such as semantic browsing, smart question answering and ontological browsing (Stojanovic *et al*., 2001; Studer *et al*., 2002; Moreale and Vargas-Vera 2003).

The paper is organized as follows: Section 2 describes the AQUA process model. Section 3 describes a comprehensive set of components used in AQUA such as Query Logic Language (QLL); the AQUA query satisfaction-algorithm; the similarity algorithm embedded in AQUA; an evaluation of our similarity algorithm; failure analysis; output enhancements and an evaluation using queries relevant to the AKT ontology. Section 4 presents AQUA as a open-domain question answering system. It describes the query classification component and learning patterns and extraction of answers using a library of patterns. Section 5 describes related work closest in spirit to AQUA. Finally, section 6 gives conclusions and directions for future work.

[3] The AKT ontology contains classes and instances of people, organizations, research areas, publications, technologies and events (http://akt.open.ac.uk/ocml/domains/akt-support-ontology/).

2. AQUA process model

The AQUA process model generalizes other approaches by providing a uniform framework which integrates NLP, Logic, Ontologies and information retrieval. Within this work we have focused on creating a process model for the AQUA system. Figure 1 shows the architecture of our AQUA system.

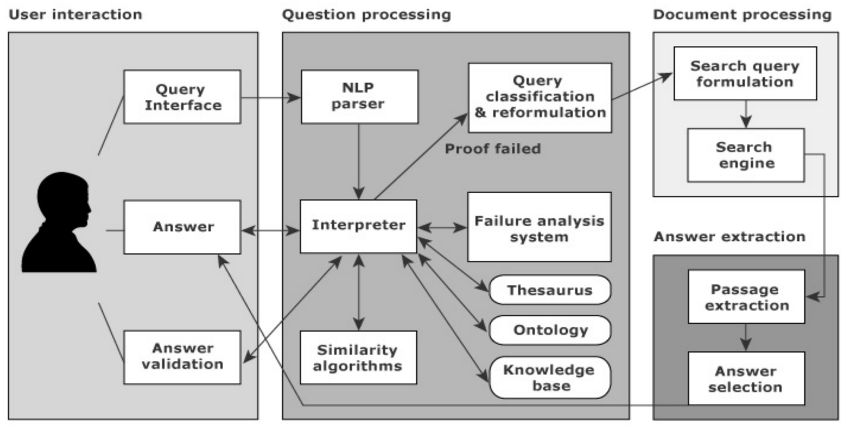


Figure 1. The AQUA architecture

In the process model there are four phases: **user interaction, question processing, document processing and answer extraction.**

1. **User interaction**. The user inputs the question and validates the answer (indicates whether it is correct or not). This phase uses the following components:

   - *Query interface*. The user inputs a question (in English) using the user interface -a simple dialogue box. The user can reformulate the query if the answer is not satisfactory.

   - *Answer*. A ranked set of answers is presented to the user.

   - *Answer validation*. The user gives feedback to AQUA by indicating agreement or disagreement with the answer.

2. **Question processing**. Question processing is performed in order to understand the question asked by the user. This 'understanding' of the question requires several steps such as parsing the question, representation of the question and classification. The question processing phase uses the following components:

- *NLP parser*. This segments the sentence into subject, verb, prepositional phrases, adjectives and objects. The output of this module is the logic representation of the query.

- *Interpreter.* This finds a logical proof of the query over the knowledge base using unification and resolution algorithms.

- *WordNet/Thesaurus.* AQUA's lexical resource.

- *Ontology*. Currently AQUA works with a single ontology – the AKT reference ontology which contains people, organizations, research areas, projects, publications, technologies and events. But in future the AQUA architecture will have a mediator which will select ontologies relevant to a Query. This is ongoing work at the Knowledge Media Institute (KMi).

- *Failure-analysis system*. This analyzes the failure of given question and gives an explanation of why the query failed. Then the user can provide new information for the pending proof and the proof can be re-started. This process can be repeated as needed.

- *Question classification & reformulation*. This classifies questions as belonging to any of the types supported in AQUA, (what, who, when, which, why and where). This classification is only performed if the proof failed. AQUA then tries to use an information retrieval approach. This means that AQUA has to perform document processing and answer extraction phases.

3. **Document Processing**. A set of documents are selected and a set of paragraphs are extracted. This relies on the identification of the focus[4] of the question. Document processing consists of two components:

    - *Search query formulation*. This transforms the original question, **Q** into a new question **Q'**, using transformation rules. Synonymous words can be used, punctuation symbols are removed, and words are stemmed.

    - *Search engine*. This searches the web for a set of documents using a set of keywords.

4. **Answer processing**. In this phase answers are extracted from passages and given a score, using the two components:

    - *Passage selection*. This extracts passages from the set of documents likely to have the answer.

    - *Answer selection*. This clusters answers, scores answers (using a voting model), and lastly obtains a final ballot.

3. AQUA as closed-domain QA

3.1 Query Logic Language (QLL)

In this section we present the Query Logic Language (QLL) used within AQUA for the translation of the English question into its Logic form. In QLL variables and predicates are assigned types. Also, QLL allows terms (in the standard recursively-defined Prolog sense (Clocksing et al. 1981, Lloyd 1984). AQUA uses QLL as an inter-media language as is shown in example (section 5).

[4] Focus is a word or a sequence of words which defines the question and disambiguates it in the sense that it indicates what the question is looking for.

Like Prolog or OCML (Motta 1999), QLL uses unification and resolution (Lloyd 1984). However, in the future we plan to use Contextual Resolution (Pulman 2000). Given a context, AQUA could then provide interpretation for sentences containing contextually dependent constructs.
Again like Prolog QLL uses closed-world assumption. So facts that are not **provable** are regarded as **false** as opposed to **unknown**. Future work needs to be done in order to provide QLL with three-valued logic. When this is done an evaluation of a predicate could produce *yes, no* or *unknown* as in Fril (Baldwin et al. 1995). Finally, QLL handles negation as failure but it does not use cuts.

Translation rules are used when AQUA is creating the logical form of a query, i.e. from grammatical components into QLL. The set of translation rules we have devised is not intended to be complete, but it does handle all the grammatical components produced by our parser. Note that variables are denoted by strings starting with a **?**, for example, **?t**. The form of the logical predicates introduced by each syntax category is described as follows:

- **Nouns** (**without complement**) introduce a predicate of arity 1. For example the noun capital introduces the predicate capital (?x :type ?t_1) which restricts the type of value ?x to be the name of the city.

- **Nouns** (**with complement**) introduce a predicate of arity equal to the number of complements plus one. The pattern for n complements is as follows:

pred_name(? argument$_1$: type ?t_1, ...,? argument$_n$: type ?t_n, ? argument $_{n+1}$: type ?t $_{n+1}$).

For example, in the question ``What is the population of the UK?" the noun population is translated into the predicate:

population(uk: ?type t_1, ?x: type ?t_2).

- **Qualitative adjectives** introduce a predicate of arity 1. For example, the adjective ``AKT technology" translates into

akt_technology(?x : type ?t_1).

- **Quantitative adjectives** introduce a binary predicate. For example, the question ``How big is London?" translates into the following predicate:

has-size(london: type ?t_1, ?t :type ?t_2).

- **Prepositions** introduce a binary predicate. The pattern is as follows:

name_preposition(?argument$_1$: type ?t_1, ? argument$_2$: type t_2).

For example, the preposition *between* gets translated in the predicate:

between(?x : type t_1, ?y : type t_2).

- **Verbs** introduce predicates with one or more arguments. The first argument should be the subject of the verb, the second is the direct object, the third is the indirect object (if any) and complements (if any). For example, ``*David Brown visited KMi?*" is translated into the following predicate:

visited(david_brown: type ?t_1 , kmi: type ?t_2).

A set of built-in predicates is also available in QLL. These include length, max, min, greater_than,less_than.

3.2 The AQUA query-satisfaction algorithm

This section presents the main algorithm implemented in the AQUA system. For the sake of space, we present a condensed version of our algorithm. In this following algorithm AQUA uses steps 1-4.1 to evaluate query over the populated AKT reference ontology. Steps 4.2 to 5 are used by AQUA trying to satisfy the query using the Web as resource.

1. Parse the question into its grammatical components such as subject, verb, prepositions, object and so on.

2. Use the ontology to convert from the language of QLL to a standard predicate logic. The ontology is used by a pattern-matching algorithm[5] to instantiate type variables, and allow them to be replaced with unary predicates.

3. Re-write the logic formulae using our similarity algorithm (described in the next section)

4. Evaluate/execute the re-written logic formula over the knowledge base.

4.1 **If** the logic formulae is satisfied, **then** use it to provide an answer

4.2 **else**

- Classify the question as one of the following types:
    - **what** - specification of objects, activity definition
    - **who** - person specification
    - **when** - date
    - **which** - specification of objects, attributes
    - **why** - justification of reasons
    - **where** - geographical location

- Transform the query **Q** into a new query **Q'** using the important keywords.

- Launch a search engine such as Google[6] with the new question **Q'**. AQUA will try to satisfy the user query using other resources such as the Web.

- Analyze retrieved documents which satisfy the query **Q'**.

- Perform passage extraction.

- Perform answer selection.

- Send answer to user for validation.

[5] The pattern-matching algorithm tries to find an exact match with names in the ontology.
[6] http://www.google.com

5. Stop.

The example below shows just one aspect of use of ontology in question answering: ‘ontology traversal’ (i.e. generalization/specialization). Suppose we have the question:

Which technologies are used in AKT?'

AQUA tries to answer question as follows.

The English question is translated into the QLL expression

use(?x : type technology, akt : type ?y)

which is then converted – using the AKT ontology - to the standard (Prolog-style) expression by using the AKT ontology

$\exists$ X:Domain   technology(X) & project(akt) & use(X,akt).

where the meaning of the quantifier is determined by the type of the variable it binds. For our example, let us decide that **Domain** is the set containing the union of of each of the following congruence classes: people, projects, organizations, research areas, technologies, publications and events.

If there is a technology in the AKT project which is defined in the knowledge base then X will be bound to the name of the technology. Let us imagine the scenario where instances of technology are not defined in the AKT reference ontology. However, AQUA found in the AKT reference ontology that the relation ‘commercial_technology’ is a subclass of ``technology". Then commercial_technology is a particular kind of technology, i.e.

commercial_technology $\subseteq$ technology

By using the subsumption relation our initial formula is transformed into the following one:

$\exists$ X:commercial_technology   commercial_technology(X) & project(akt) & use(X,akt).

AQUA then tries to re-satisfy the new question over the knowledge base. This time the question succeeds with X instantiated to the name of one of the AKT technologies.

We can see from the algorithm that AQUA tries to satisfy a user query using several resources. Future implementations of AQUA could benefit from using the results obtained by the Armadillo[7] (Ciravegna et al. 2003) information extraction engine running in the background as a complementary knowledge harvester. For instance, the user could ask the question ‘What publications has Yorik Wilks produced?’ This is a good example of a query for which AQUA could make a request to Armadillo. Armadillo would

[7] Armadillo is an information extraction engine which uses resources such as CiteSeer to find a limited range of information types such as publications.

find the publications of Yorik Wilks (for example, using CiteSeer[8] or his personal web site). Next, Armadillo would parse documents, retrieve the requested information and pass the set of publications back to AQUA to render an appropriate answer.

3.3 Concept and relation similarity algorithm

Similarity ha been an important research topic in several fields such linguistic, Artificial intelligence (in particular in the field of Natural Language Processing and Fuzzy Logic). The range of application of measure of similarity ranges from word sense disambiguation, text summarization, information extraction and retrieval, question answering, automatic indexing and automatic correction of codes.
We found that there are two types of similarity: syntactically and semantic similarity. Syntactic similarity can be defined as functions over terms. For instance the hamming distance (used in Information Theory). This similarity is defined as the number of positions with different characters in two terms with the same length. Whilst semantic similarity can be defined as Miller and Charles (Miller et al 1991) as a continuous variable that describes the degree of synonymy between two words.

When evaluating similarity in a taxonomy the most natural way to access similarity is to evaluate the distances between the two concepts being compared. Therefore the shorter is the path from one to another means that they are more similar. This approach has been used as measure of similarity. However one of the main drawbacks is that it relies on the notion that links in a taxonomy represent uniform distances (Resnik 1995; 1998). Our own view is that similar entities are assumed to have common features[9]. For instance (university, research_institute) but it is also the case that dissimilar entities may also be semantically related by the relation meronym or holonym such as (student –person; bicycle-wheel) .

Our similarity algorithm assess concept similarity and relation similarity. It compares extended graph obtained from the user query (plus informative classes from ontology) and a graph which represents a subset of the ontology (relevant to the query). As an inter-media stage our it creates the intersection upon nodes (described in section 3) between the two graphs (the **graph of the query** and the **graph obtained from the ontology)**. Then, our similarity algorithm assess concept similarity and relation similarity using the Dice Coefficient using the most informative classes from the ontology.

The success of the attempt to satisfy a query depends on the existence of a good mapping between the names of relations used in the query and names of relations used in the knowledge base/ontology. Therefore, we have embedded in AQUA a similarity algorithm. Our similarity algorithm uses both ontological structures and instances in the selected ontology, the Dice coefficient and the WordNet thesaurus. Our similarity algorithm differs from other similarity algorithms in that it uses ontological structures and also instances. Instances provide evidential information about the relations being analyzed. This is an important distinction between the kind of similarity which can be achieved using only WordNet or the similarity which can be calculated using distances to super-classes (Wu et al. 1994). In the former, WordNet brings all the synsets found even the ones which are not applicable to the problem being solved. Whilst in the latter one, the idea of a common super-class between concepts is required. In our case, it is not a necessary condition.

We present an explanation of our algorithm when arguments in the user query are grounded (instantiated terms) and they match exactly (at the level of strings) with instances in the ontology. A detailed description of the algorithm and example can be found in (Vargas-Vera *et al*. 2003a, 2003b, 2004).

[8] http://citeseer.nj.nec.com/cs
[9] The common features in two entities might not be so discriminative as the features which are different in them.

The algorithm uses grounded terms in the user query. It tries to find them as instances in the ontology. Once they are located a portion of the ontology (G_2) is examined including neighborhood classes. Then, using knowledge from the ontology, an intersection [10] (G_3) is found between augmented query G_1 and G_2 to assess structural similarity. It could be the case that in the intersection G_3 several relations can include the grounded arguments. Then the similarity is computed for all relations (containing elements of the user query) using the Dice Coefficient. Finally, the relation with the maximum Dice Coefficient value is selected as the most similar relation.

AQUA reformulates the query using the most similar relation and it then tries to prove the reformulated query. If no similarity is achieved using our similarity algorithm then AQUA provides the user with the synsets obtained from WordNet. From this offered set of synsets, the user selects the suitable one.

The similarity algorithm for relations is defined as follows:

SimilarityBase algorithm:

Case 1: X_1 and X_2 are grounded arguments.

1. Translate the question to First Order Logic i. e. predicate_name(X_1, X_2)

2. $\exists$ relation connecting C_1 and C_2 $\Lambda \exists C_1 \supset X_1 \Lambda \exists C_2 \supset X_2$ such that

relation(C_1,C_2) where relation is an ontological relation between C_1 and C_2.

3. $\exists$ relation connecting C_1 and C_2 $\Lambda \exists C_1 \supset X_1 \Lambda \exists C_2 \supset X_2$ such that

relation(C_1, C_2) $\Lambda \exists S_1 \supset (U_{11} \subset U_{12} \subset \dots \subset U_{1n}) \supset C_1 \Lambda \exists S_1 \Lambda (U_{11} \subset U_{12} \subset \dots \subset U_{1n}) \supset C_2$

4. $\exists$ relation connecting C_1 and C_2 $\Lambda \exists C_1 \supset X_1 \Lambda \exists C_2 \supset X_2 \Lambda S_1 \supset X_1 \Lambda S_2 \supset X_2 \Lambda \exists (U_{11} \subset U_{12} \subset \dots \subset U_{1n}) \subset C_1 \Lambda \exists (U_{21} \subset \dots \subset U_{2n}) \subset C_2$
where Uij is a subclass of U_{ij+1}

5. Find the intersection, G_3, of G_1 and G_2 based upon the node labels.

6. Let be A and B vectors containing the features used to compare similarity.

Compute Concept _similarity = 0 no common concepts

Concept _similarity = 1 same set of concepts , otherwise

Concept_similarity = sim_dice(A,B) = $2*\sum_1^n aibi / \sum_1^n ai^2 + \sum_1^n bi^2$
vectors A and B are filled with the number of concept nodes of graph G_1 and G_2 respectively.

7. Compute Relation_ similarity = d_i = sim_dice(A,B) = $2*\sum_1^n aibi / \sum_1^n ai^2 + \sum_1^n bi^2$
vector A and B are filled with the number of arcs in the immediate neighborhood of the graph G_1 and G_2 respectively.

[10] Intersection means to find a sub-graph in G_2 which contains all concepts contained in graph G_1 using subsumption relation.

8. maximum(d_i) where i=1,n

The algorithm builds graphs from the user query G_1, obtains a fragment from the ontology containing the relevant nodes G_2 and builds an intersection (G_3) between G_1 and G_2.
Step 2 describes the construction of the graph G_1 for the query. This is created using subject (X_1), relation, object (X_2) and the most representative classes for X_1 and X_2 respectively.
Step 3 describes how the algorithm finds sub-hierarchy containing grounded[11] arguments/concepts from the user question (i.e., the neighborhood containing the grounded arguments).
Step 6. the similarity between G_1 and G_3 is computed using the Dice coefficient. The vectors A and B are filled with the number of concept nodes of graph G_1 and G_3 respectively then

Concept _similarity = 0 no common concepts

Concept _similarity = 1 same set of concepts , otherwise

Concept _similarity $= 2*\sum_1^n aibi / \sum_1^n ai^2 + \sum_1^n bi^2$

Step 7. the similarity between G_1 and G_2 is computed using the Dice coefficient.

Relation_similarity $= d_i = 2*\sum_1^n aibi / \sum_1^n ai^2 + \sum_1^n bi^2$

vector A and B are filled with the number of arcs in the immediate neighborhood of the graph G_1 and G_2 respectively.

A procedure called SimilarityTop uses the SimilarityBase algorithm (defined above), WordNet synsets, and feedback from the user. The SimilarityTop procedure checks if there is similarity between the name of the relation/concept in the query and the relation/concept in the selected ontology. If there is no similarity, then it offers the users all the senses which are found in the WordNet thesaurus. It is SimilarityTop that AQUA uses, with the selected sense, to rewrite the logic formulae. The main steps are defined as follows:

SimilarityTop procedure:

- Call our **SimilarityBase** algorithm (defined above)

- If ontological_relation $\neq$ null_string then

  evaluate_query(ontological_relation(Arg_1,Arg_2))

- Else
  - o Obtain synsets for relation_question using WordNet thesaurus
  - o Ask user to select sense from the ones that WordNet thesaurus provided
  - o Call evaluate_query(selected_sense(Arg_1, Arg_2))

To illustrate how our similarity algorithm works we present an example.

[11] grounded argument means instantiated argument.

3.3.1 Working Example

In this section we illustrate how our similarity algorithm works by presenting a working example. Note that, in more complex examples, the graph G_2 could have several relations which could be assessed for similarity.

Let us imagine that someone asks the question: "Does Enrico Motta work on AKT?"
work(enrico-motta,akt).

By refining our query using the AKT reference ontology we obtain the following formula:

project(akt) & researcher(enrico-motta) & work(enrico-motta,akt).

If AQUA evaluates this query over the knowledge base, it is likely that the question will fail. The problem is that the name of the relations in the knowledge base and the names of relations in the question might be completely different. AQUA will ask the user if they want to use similarity. If the answer to this question is ``yes" then AQUA builds three graphs: the graph associated with the question (G_1), the graph using ontological structures (G_2) and the intersection graph of G_1 and G_2. These graphs are shown in Figure 2.

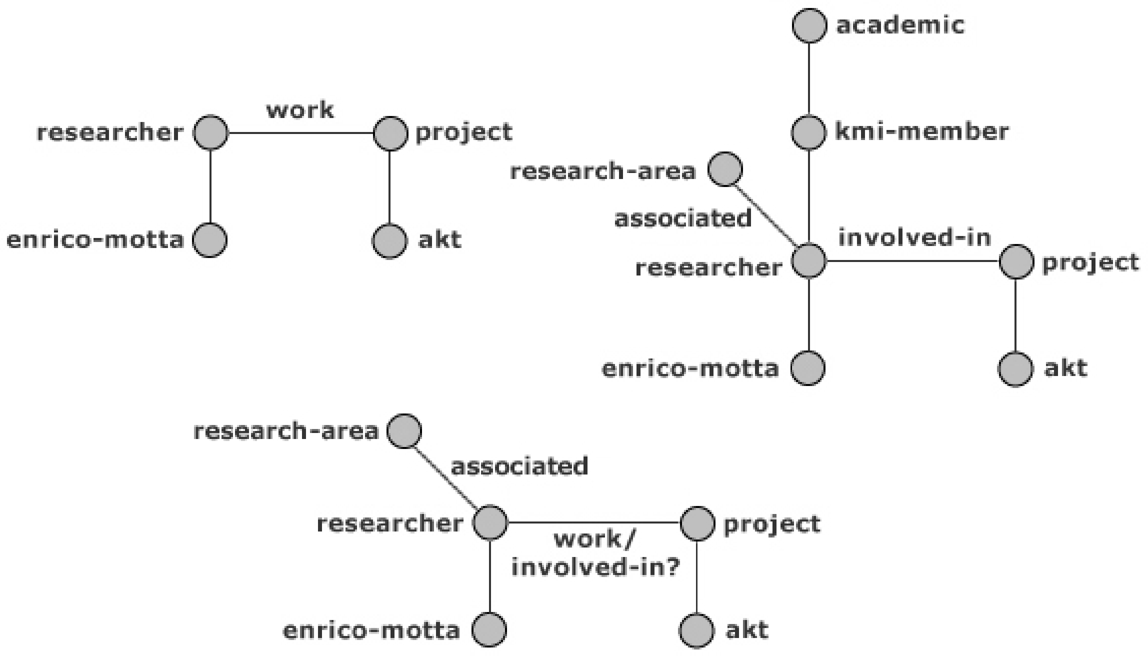


Figure 2. Graph G_1 and G_2 and intersection G_3

The relation similarity is computed as follows:
In the example shown in Figure 2 using G_1 and G_3.

Concept _similarity = 1

In order to compute Relation_similarity we use G_1 and G_2 graphs and vectors A=(2,2) and B=(3,2)

Then, Relation_similarity = $d_i = 2*\sum aibi / \sum_1^2 ai^2 + \sum_1^2 bi^2$

$d_i = (A,B) = (2 *(6+4)) / (4+4) +(9+4) = (2*10)/21 = 20/21=0.9$

The outputs of the similarity algorithm is 0.9 and the name of the relation between concept *researcher* and concept project is "involved-in", Then the question is re-written as follows:

project(akt) & researcher(enrico-motta) & involved-in(enrico-motta,akt).

The question is re-evaluated by the interpreter. By using the similarity algorithm AQUA tries to reduce to a minimum the possibility of failure because of mismatches between relation names. Since there are several techniques for assessing similarity (Doan et al. 2002, Noy et al. 2000, Wu et al. 1994), a future implementation of AQUA will contain several algorithms for similarity.

3.3.2 Similarity algorithm evaluation

Some of the typical queries used in our experiment are shown in Table I. This table shows the concept and relation similarity values obtained using **SimilarityBase algorithm**. The evaluation has been performed using the AKT reference ontology which describes academic life.

Question number	**NLP query**	**Query expressed as logic predicate**	**Reformulated predicate using SimilarityBase algorithm**	**Concept Similarity value**	**Relation Similarity value**
1	Does Enrico Motta work in akt?	work(enrico-motta, akt) & project(akt)	involved (enrico-motta,akt) & project(akt)	1	0.9
2	Does Maria Vargas-Vera is employed by akt?	employed(maria-vargas-vera,akt) & project(akt)	involved (maria-vargas-vera,akt) & project(akt)	1	0.9
3	Is David Celjuska associated to akt?	associated(david-celjuska,akt) & project(akt)	involved(david-celjuska,akt) & project(akt)	1	0.9
4	Does Peter Scott works in Semantic Web and in akt?	work(peter-scott,semantic_web) & research_area(peter-scott,semantic_web) & work(X,akt) & project(akt)	involved(peter-scott,semantic_web) & research_area(peter-scott,semantic_web) & involved(peter-scott,akt) & project(akt)	1	0.9

Table I. query and its reformulation using the **SimilarityBase** algorithm

Question 3 can be reformulated as suggested in Table I by using our **SimilarityBase algorithm**. However if we use the suggested WordNet senses as first resource then the query can be reformulated as is shown Table II. The latest is not an optimal solution for AQUA since it will try to evaluate (as logic predicates) each of the reformulation. Therefore, our goal of having a question answering system in real time it might suffers of the problem of slow response time. The question 3 example (from Table I) illustrates that by using a domain specific ontology queries can have less alternative reformulations.

Query	**Reformulations using WordNet senses**

Is David Celjuska associated to akt?	linked(david-celjuska,akt) & project(akt)
	related(david-celjuska,akt) & project(akt)
	connected(david-celjuska,akt) & project(akt)
	affiliated(david-celjuska,akt) & project(akt)
	linked-up(david-celjuska,akt) & project(akt)

Table II. query and its reformulation using WordNet suggested senses

When AQUA fails to find a candidate for similarity using SimilarityBase (i.e. null value is returned). Then ,it tries to find possible reformulations using the synsets of WordNet. Table III shows an examples of queries and possible reformulations offered to the user by AQUA.

3.4. Failure Analysis

A question is evaluated over the knowledge base. If the evaluation fails then the system must provide an explanation for why the question fails? There are several reasons why a query evaluated over the knowledge base could fail:

- The **term/concept** is not defined in the knowledge base. Failure occurred because the noun in the English question was not defined in the ontology and also AQUA did not find any similarity with any of the terms/concepts in the ontology.
- One or more **relations** are not defined in the ontology (i.e. the similarity algorithm did not find similar relations).
- A **predicate** evaluation fails. The reason could be that the predicate used in translation is not a built-in predicate in QLL.

When a goal fails the system keeps a record of the failed goal. This helps AQUA to produce a suitable failure message. AQUA also has facilities for analyzing and explaining proofs. The explanation is provided both in pseudo natural language and as a visualization. In this way the user might be able to re-write the English question using only entity definitions which are already defined in the ontology and predicates defined in QLL.

3.5 Output enhancements

Figure 3 and Figure 4. show snapshots of the AQUA system. In Figure 5 we can see that AQUA not only provides a set of names which satisfy the query, but enhances its answer using information from the AKT reference ontology. It provides more information about each element in the set. For instance, it brings additional contextual information such as ‘‘AKT is a project at KMi’’ and ‘‘each person of the AKT team is a researcher at KMi’’.

Validation of answers is a difficult task in general. Therefore, AQUA provides a visualization of proofs for supporting validation of answers. Another way, provided by AQUA, to help users in validation, could be by enhancing answers with extra information. There is ongoing work at KMi on this problem. We use Magpie (Domingue et al. 2003) in this task of enhancing answers. Magpie is a semantic browser which brings information about ontological entities. For instance, it can retrieve home pages related to AKT or personal web pages of researchers working in the AKT project. All this extra information could be used by our AQUA users in answer validation.

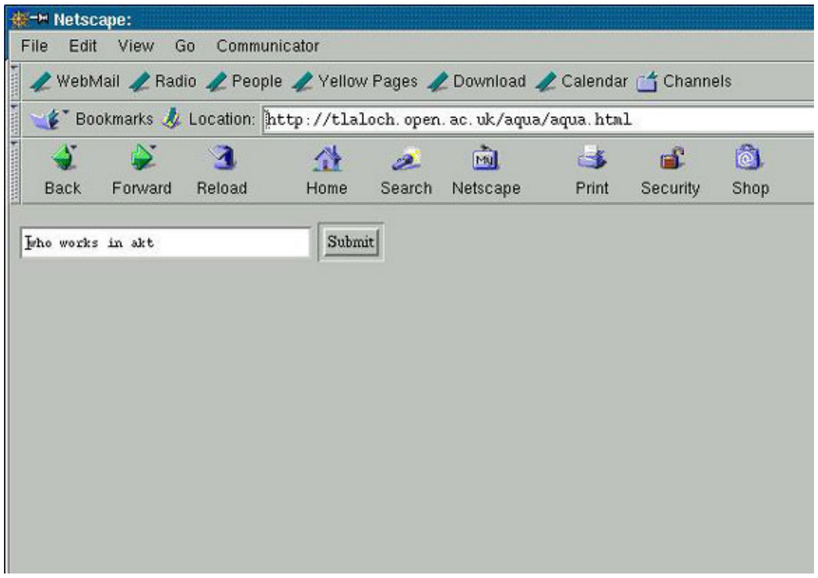
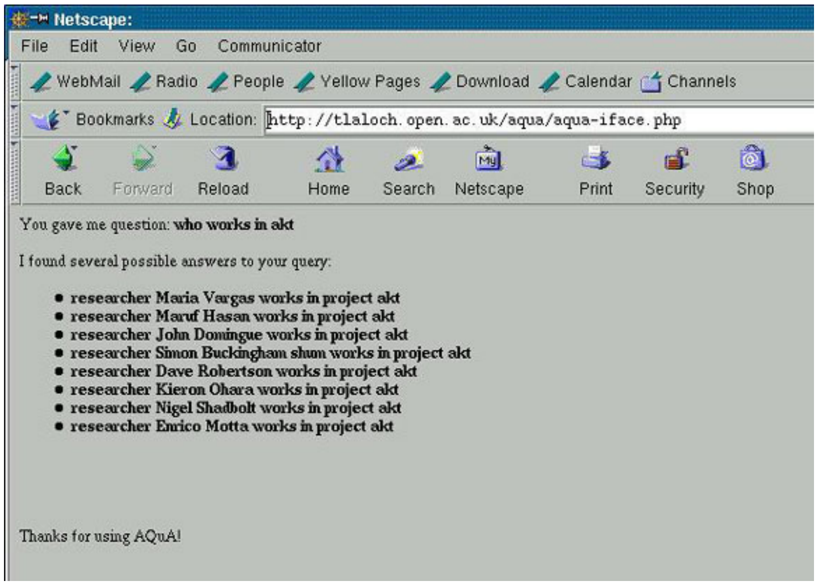

Netscape:
File Edit View Go Communicator
WebMail Radio People Yellow Pages Download Calendar Channels
Bookmarks Location: http://tlaloch.open.ac.uk/aqua/aqua.html
Back Forward Reload Home Search Netscape Print Security Shop
Who works in akt
Submit

Figure 3. The question “Who works on akt?’’ is provided to AQUA


Netscape:
File Edit View Go Communicator
WebMail Radio People Yellow Pages Download Calendar Channels
Bookmarks Location: http://tlaloch.open.ac.uk/aqua/aqua-iface.php
Back Forward Reload Home Search Netscape Print Security Shop

You gave me question: **who works in akt**

I found several possible answers to your query:

- **researcher Maria Vargas works in project akt**
- **researcher Maruf Hasan works in project akt**
- **researcher John Domingue works in project akt**
- **researcher Simon Buckingham shum works in project akt**
- **researcher Dave Robertson works in project akt**
- **researcher Kieron Ohara works in project akt**
- **researcher Nigel Shadbolt works in project akt**
- **researcher Enrico Motta works in project akt**

Thanks for using AQuA!

Figure 4. AQUA's answers to a question

3.6. AQUA Evaluation as closed-domain QA

We conducted experiments using 43 questions related to the AKT ontology. AQUA was able to answer them correctly. However, one of the limitations of the current AQUA implementation was that the parser needs to be extended in order to cope with misspelling errors and unknown verbs. A solution to the latter (unknown verbs) could be to use the approach followed in Camille (the contextual Acquisition mechanism for incremental lexeme learning (Hasting 1994,1995,1996), where the meaning of verbs is inferred from context.

Some of the typical queries used in the experiment are shown in Table III. However, a more comprehensive set of questions used in our experiment can be found at the AQUA home page: http://kmi.open.ac.uk/projects/akt/aqua/

Some questions are answered by AQUA by performing statistical analysis of the data contained in the knowledge base. Examples of such questions are, *Which researchers work on most projects?*
Do most researchers work on most projects?

Question type	**Answers**
Who	
Who works in akt?	A set of people who works on the akt project
Who owns webonto?	Name of institution
Who is John Domingue?	Senior Research Fellow KMi deputy director

What	
What is MnM?	KMi tool
What is akt	Project
Which/What	
Which technologies KMi had produced?	A set of technologies
Which researchers do not work on the akt project?	A set of researchers except researchers working on the akt project.
Which researchers work on most projects?	A set of researchers who work on almost all projects in Kmi
Which tools support the technique of Information Extraction?	A set of KMi tools making use of Information Extraction.
Which researchers work on all projects?	A set of researchers working in all Kmi projects
Which researchers work on few projects?	A set of researchers working in less than 1/3 of KMi projects
Which researchers do not work on most projects?	A set of researches working on few projects
Which publications has KMi published?	A list of publications
Where	
Where is researcher Maria Vargas-Vera based?	KMi
Type Yes/No questions:	
Do all researchers work on the akt project?	No.
Do most researchers work on most projects?	Yes
Were all technologies produced by the institution known as KMi?	No
Does Enrico Motta work on the akt project?	Yes

Table III. AQUA has been evaluated with the following category of questions and using the AKT ontology.

Further work is planned in the direction of handling a wider range of question. For example the *when* and *why* type described in section 4.

4. AQUA as open-domain QA

4.1 Query Classification

Query classification is performed in the case that the proof fails, and gives information about the kind of answer AQUA should expect. Therefore, query classification plays an important role in the case that the answer needs to be found from the Web, and cannot be satisfied using just the populated ontology. For example, given the query ``What is the capital of Mexico?'' AQUA will not find anything from the ontology, and will instead a generate keyword search and pass it to the search engine.

The classification phase involves processing the query to identify the category of answer that the user is seeking. The classification is performed using the information obtained during segmentation of the sentence. During segmentation, the system finds nouns, verbs, prepositions and adjectives. The categories of possible answers are listed below.

- what/which - this kind of question appears with a head noun that describes the category of the entity involved. For this category the head noun is extracted and WordNet is used to perform the mapping between head noun and category.
- who, whom - the category of the answer is 'person'.
- when - the category of the answer is 'date'.
- why - the category of the answer should be 'reason'.
- where - the category of the answer is 'location'.

The range of answers provided by AQUA - our question answering system - varies from yes/no answers, true/false answers, and answers consisting of a word or a sentence. In some cases, questions could have more than one answer, whilst in other cases the system might not find the answer. Currently, AQUA handles six categories of questions. Table IV shows categories of queries that AQUA is able to handle. These categories are based on the conceptual categories used in Lehnert's QUALM system (Lehnert 1978).

Question categories	**Specification**	**Examples**
Verification	Is a fact true?	Does Maria Vargas-Vera work on akt? Does D. Fensel work on akt?
Definition	What is X?	What is akt? What is ontoweb ?
Quantification	How many	How many researchers work on akt? How many people work in artificial intelligence?
Reason	Why X?	Why the sky is blue?
Concept	Who? What? When? Where	Who is D. Fensel?
Directive	Give me X	Give me Microsoft technologies Give me KMi technologies

Table IV shows a categorization of queries which can be handled using AQUA

Several answers can be correct but, in a given context, some of them can be useless. To illustrate this example we took an example from Lehner " *Where is Derfield Illinois?*'' (Lehnert 1978).

A set of possible answers are as follows:

1. Answer: 87^o 54' longitude 42^o 12' latitude.
2. Answer: Near to Lake Michigan, about 20 miles north of Chicago.
3. Answer: Next to Highland Park.
4. Answer: On the planet Earth.

For example, answer 4 ('*On the planet earth'*) is not a meaningful answer to someone who wants to travel to *Derfield Illinois*. Similarly, answer 3 is useless if the person who is asking the question does not know where *Highland Park* is. Answer 1 is appropriate if a technical specification is required.

4.2 Learning patterns

When the query cannot be satisfied using the AKT ontology (knowledge base), AQUA tries to satisfy the query using the web as a resource. In the downstream part of the AQUA architecture, we used information extraction and Machine Learning techniques. AQUA contains a set of learnt patterns from a predefined domain. As a first instance, we learnt patterns using as corpus an electronic news stories archive that we have available in our institution (Knowledge Media Institute). An example of a new story in our corpus is shown below. The title is **The AKT begins**

The AKT begins.....KMI awarded £1.2M by EPSRC

Enrico Motta
01.03.00
"*KMi has been awarded £1.2M by the UK's Engineering and Physical Sciences Research Council to carry out research in the application of knowledge technologies to support knowledge creation and sharing in organizations. This highly prestigious award has been obtained in the context of the EPSRC Programme on Interdisciplinary Research Collaborations centred on Information Technology*''

We have used UMass tools (Marmot , Badger and Crystal (Riloff:96)) for information extraction. Our answer extraction system, as most information extraction systems, uses some form of partial parsing to recognize syntactic constructs without generating a complete parse tree for each sentence. Such partial parsing has the advantages of greater speed and robustness. High speed is necessary to apply the information extraction to a large set of documents. The robustness achieved by allowing useful work to be done from a partial parsing is essential to deal with unstructured and informal texts.
The nature of the domain is very important in determining which kind of patterns are necessary in our library of patterns. Currently, in order to apply our system to a new domain the entire knowledge process of defining templates has to be repeated.

4.2.1 Learning process model

Within this work, we have focused on creating a generic process model for learning patters for the AQUA system. In the learning phase, we have devised three activities : markup, learning and extraction. We will provide more details of each of the activities in turn.

Markup

In markup phase, we used half of the corpus (100 news stories). We trained the system using three categories of news stories: visits, awards and academic conferences. We annotated text documents (written in plain ASCII or HTML) with a set of tags. These tags are semantic annotations to relevant entities in the text. During the markup phase as the text is selected the system inserts the relevant SGML tags into the document. This markup was done using the TMI interface provided with the UMass tools.

Learning

This phase was implemented by integrating two components: Marmot and a learning component called Crystal, both from UMass . We describe them briefly in this paper. However, full descriptions can be found in (Riloff 1996a)

Marmot is a natural language preprocessing tool that accepts ASCII files and produces an intermediate level of text analysis that is useful for information extraction applications. Sentences are separated and segmented into noun phrases, verb phrases prepositional phrases. Marmot has several functionalities: it preprocesses abbreviations to guide sentence segmentation, resolves sentences boundaries, identifies parenthetical expressions, recognizes entries from a phrasal lexicon and replace them, recognizes dates and duration phrases, performs phrasal bracketing of noun, preposition and adverbial phrases, and finally scopes conjunctions and disjunctions.

Crystal is a dictionary induction tool. It derives a dictionary of concept nodes from a training corpus. The first step in dictionary creation is the annotation of a set of training texts by a domain expert. Each phrase that contains information to be extracted is tagged (with SGML style tags).
Crystal initializes a concept nodes dictionary for each positive instance of each type of event. The initial concept node definitions are designed to extract the relevant phrases in the training instance that creates them but are too specific to apply to unseen sentences. The main task of Crystal is to gradually relax the constraints on the initial definitions and also to merge similar definitions. Crystal finds generalizations of its initial concept node definitions by comparing definitions that are similar. This similarity is deduced by counting the number of relaxations required to unify two concept node definitions. Then a new definition is created with constraints relaxed. Finally the new definition is tested against the training corpus to insure that it does not extract phrases that were not marked with the original two definitions. This means that Crystal takes similar instances and generalizes into a more general rule by preserving the properties from each of the concept node definitions which are generalized.

Extraction

A third component called Badger (from UMass) was also integrated into our event recognition system. Badger makes the instantiation of templates: its main task is to take each sentence in the text and see if it matches any of our concept node definitions. If no extraction concept node definition applies to a sentence, then no information will be extracted; this way irrelevant text can be processed very quickly.

9.2.2 Examples of extracted patterns using KMi news stories as training corpus

Each event in our system has several patterns which can be used to recognize it (e.g. see questions of the **visitor , academic-conference and award** types). The patterns are shown in Table V.

KMi News stories Corpus

Visits patterns	**Questions**
X visited Y	Who visited Edinburgh University?
Y was visited by X	Does OU was visited by Mr Blair ?
Y visited by X on Z	Who visited the OU on 12 July 2003?
Y were visited by X	Who were visited by Mr Davis?
Visit to Y	
X came	Who came?
X came to visit Y	Who came to visit The Open University?
Y hosted X	Who hosted Ernesto Compatangelo?
Y hosted a visit from X	Who hosted a visit from Ernesto Compatangelo?
Y had a visit from X	Who had a visit ?
Y welcomes X	Who welcomes Gilliam Vincent?
In all patterns shown above X is a person, Y is a place/institution and Z is a location.	
Awards patterns	
ORG has been awarded MONEY by FUNDER	Who was awarded 10000 Euros by the European Commission ?
ORG has been received MONEY from FUNDER to carry out research in TOPIC	Which organization has been awarded 10000 Euros from the EC to carry out research in the semantic web?
ORG has been awarded MONEY by FUNDER to carry out research in TOPIC	Which organization has been awarded 10000 Euros by EC to carry out research in the semantic web?
ORG has been awarded MONEY on TOPIC	Which organization has been awarded 10000 Euros by EC to carry out research on the semantic web?
awards MONEY to ORG	
FUNDER has awarded MONEY to ORG	What funder has awarded 10000 Eu to Kmi?
brings MONEY to ORG	Who brings money to ORG?
Academic conference patterns	
held at PLACE	What was held at the Open University?
taking place in PLACE	Where did the meeting take place?
hosted by ORG	Who hosted the conference?
organized by ORG	Who organized the conference?
paper entitled TITLE	What was the title of the paper?

Table V shows a list of patterns and questions

The experiments showed that an automatic mechanism is needed in order to determine which extraction rules are spurious. We believe this problem can be solved by associating confidence value to the extraction rules (Vargas-Vera and Celjuska 2003c). Previous work has reported that spurious patterns were deleted manually from the library of rules under the assumption that they were not likely to be of much value (Riloff 1996a). However, our experiments reported in (Vargas-Vera and Celjuska 2003c) were carried out without deleting spurious rules.

5. Related work

There are many trends in question answering (Katz 1997, Katz et al. 1988a, Katz 1988b, Plamondon et al. 2001, Burke et al. 1997, Moldovan 1999, Hovy et al. 2001a, Hovy et al. 2001b,Attardi 2001). However, in this paper, we only describe the systems most closely related to the AQUA system philosophy.

MULDER is a web-based QA system (Kwok 2001) that extracts snippets called summaries and generates a list of candidate answers. However, unlike AQUA, the system does not exploit an inference mechanism, and so, for example, cannot use semantic relations from an ontology.

QUANDA is closest to AQUA in spirit and functionality. QUANDA takes questions expressed in English and attempts to provide a short and concise answer (a noun phrase or a sentence) (Breck et al. 99). Like AQUA, QUANDA combines knowledge representation, information retrieval and natural language processing. A question is represented as a logic expression. Also knowledge representation techniques are used to represent questions and concepts. However, unlike AQUA, QUANDA does not use ontological relations.

ONTOSEEK is an information retrieval system coupled with an ontology (Guarino 1999). ONTOSEEK performs retrieval based on content instead of string based retrieval. The target was information retrieval with the aim of improving recall and precision and the focus was specific classes of information repositories: Yellow Pages and product catalogues. The ONTOSEEK system provides interactive assistance in query formulation, generalization and specialization. Queries are represented as conceptual graphs, then according to the authors "the problem is reduced to ontology-driven graph matching where individual nodes and arcs match if the ontology indicates that a subsumption relation holds between them". These graphs are not constructed automatically. The ONTOSEEK team developed a semi-automatic approach in which the user has to verify the links between different nodes in the graph via the designated user interface. In contrast, AQUA does not require user intervention in building the **graph of the query** and the **graph obtained from the ontology.**

6. Conclusions and future work

In this paper we have presented AQUA - a question answering system which merges NLP, Logic, Information Retrieval techniques and Ontologies[12]. AQUA translates English questions into logical queries (expressed in the QLL language) that are then used to generate of proofs. Currently, AQUA is coupled with the AKT reference ontology for the academic domain. In the future, we plan to couple AQUA with a set of ontologies from our repertoire of ontologies.

[12] AQUA has been implemented in Sicstus Prolog, C, OCML and PHP.
information
Information extraction can be seen as the task of pulling predefined relations from texts.

AQUA makes use of an inference engine which is based on the Resolution algorithm. However, in future it will be tested with the Contextual Resolution algorithm which will allow the carrying of context through several related questions

We have also presented the similarity algorithm which we have embedded in AQUA: this uses Ontological structures, the Dice coefficient and WordNet synsets. This algorithm is used by AQUA to ensure that the question does not fail because of a mismatch between names of relations. In the future, we intend to provide AQUA with a library of similarity algorithms.

We are currently developing M-AQUA, which is a Question Answering System which handles multiple ontologies. It relies on a mediator which make sense of which ontology should be used for answering a query. Further research need to be done in this direction. However, preliminary results are encouraging.

One of the main restrictions of AQUA is that it only can answer questions in a isolation (i.e. it does not handle a set of follow up questions related to the initial question). Therefore, we will explore how to handle context across a set of queries.

Further work needs to be carried out to determine which data are queried most frequently by users. Then AQUA should be given patterns for such queries. We also could the improve response times for expensive queries.

Finally, future research can be focused one of the main problems found in AQUA. One such problem is that it needs to learn paraphrases to improve the QA process. For instance AQUA could learn that X killed Y has the same linguistic meaning that Y was assassinated by X (Duclaye et al. 2002).

Acknowledgments

This work was funded by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N157764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. The authors would like to thank Arthur Stutt and Mark Gaved for their invaluable help in reviewing the first draft of this paper. In particular we want to thank Gaston Burek for his help in formalizing the similarity algorithm. We also thank Martin Dzbor for his help on the AQUA interface and Gaston Burek for helping to formalize the similarity algorithm.

References

Askjeeves: http://askjeeves.com/, 2000.

Attardi G. and Cisternino A. and Formica F. and Simi M. and Tommasi A. (2001): Proceedings of TREC-9 Conference, NIST, pp 633-641, 2001.

Baldwin J. F. and Martin T.P. and Pilsworth B. W. (1995): Fril - Fuzzy and Evidential Reasoning in Artificial Intelligence. Research Studies Press in 1995.

Budanitsky A. and Hirst G. (2001) Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, June 2001.

Burke R. D. and Hammond K. J. and. Kulyukin V. A. and Lytinen S. L. and Tomuro N. and Schoenberg S. (1997): Questions answering from frequently-asked question files: Experiences with the FAQ Finder System. The University of Chicago, Computer Science Department, TR-97-05, 1995.

Brickley D. and Guha R.(2000): Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Web Consortium, URL:http://www.w3.org/TR/2000/CR-rdf-schema-20000327.

Breck E. and House D. and Light M. and Mani I. (1999): Question Answering from Large Document Collections, AAAI Fall Symposium on Question Answering Systems, 1999.

Ciravegna F. and Dingli A. , Guthrie D. and Wilks Y. (2003): Mining Web Sites Using Unsupervised Adaptive Information Extraction. Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistic, Budapest Hungary, 2003.

Clocksin W. F. and Mellish C. S. (1981): Programming in Prolog, Springer-Verlag, 1981.

Doan A. and Madhavan J. and Domingos P. and Halevy A. (2002): Learning to Map between Ontologies on the Semantic Web. In Proc. of the 11th International World Wide Web Conference (WWW2002), 2002.

Domingue J. and Dzbor M and Motta E. (2003): Semantic Layering with Magpie. The Second International Semantic Web Conference, Sanibel Island FL, USA, October 2003.

Duclaye F. and Francois Y. (2002): Using the Web as a Linguistic Resource for Learning Reformulations Automatically , Proceedings of the third international conference on language resources and evaluation (LREC'02) vol. 2, pp. 390-396, Las Palmas, Canary Islands - Spain, May 2002.

Frakes W. and Baeza-Yates R. (1992): Information Retrieval: Data Structures & Algorithms, Prentice Hall, 1992.

Guarino N. (1999): OntoSeek: Content-Based Acess to the Web, IEEE Intelligent Systems, pp 70-80, 1999.

Hastings, P. (1994): Automatic Acquisition of Word Meaning from Context. PHD. Thesis, University of Michigan, Ann Arbor, MI, 1994.

Hastings, P. (1995): Use of context in an automatic lexical acquisition mechanism. In Iwanska, L. (Ed), Proceedings of the Workshop on Context in Natural Language of the 14th International Join Conference on Artificial Intelligence, 1995.

Hastings, P. (1996): Implications of an automatic lexical acquisition mechanism. In Wermter, S ., Riloff, E., & Scheler, C.. (Eds.), Connectionist Statistical and Symbolic Approaches to Learning for Natural Language Processing. Springer -Verlag, Berlin, 1996.

Hayes P. (2002): RDF Model Theory, W3C Working Draft, URL:http://www.w3.org/TR/rdf-mt/

Hovy E. and Gerber L. and Hermjakob U. and Junk M. and Liu C-Y (2001a): Question Answering in Webclopedia, Proceedings of TREC-9 Conference, NIST, 2001.

Hovy E. and Gerber L. and Hermjakob U and Liu C-Y and Ravichandran D. (2001): Toward Semantics-Based Answer Pinpointing, Proceedings of DARPA Human Language Technology conference (HLT), 2001.

Katz B. (1997): From sentence processing to information access on the world wide web, Proceedings of AAAI Symposium on Natural Language Processing for the World Wide Web, 1997.

Katz B. and Levin B. (1988a): Exploiting Lexical Regularities in Designing Natural Language Systems, MIT Artificial Intelligence Laboratory, TR 1041, 1988.

Katz B. (1988b): Using English for Indexing and Retrieving, MIT Artificial Intelligence Laboratory, TR 1096, 1988.

Kwok C. and Etzioni O. and D. S. Weld D. S. (2001): Scaling Question Answering to the Web, World Wide Web, pp 150-161, 2001.

Lassila O. and Swick R. (1999): Resource Description Framework (RDF): Model and Syntax Specification. Recommendation. World Wide Web Consortium, URL: http://www.w3.org/TR/REC-rdf-syntax/.

Lehnert W. G. (1978): The Process of Question Answering . Lawrence Erlbaum Associates, Inc, Publishers, 1978.

Lin D. and Pantel P. (2001): Discovery of Inference Rules for Question Answering, Journal of Natural Language Engineering, 2001.

Lloyd J. W. (1984): Foundations of Logic Programming, Springer-Verlag, 1984.

Manning C. D. and Schutze H. (1999): Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge Massachusetts, 1999.

Miller G. A. and Charles W. G. (1991): Contextual correlates of semantic similarity. Language and Cognitive Processes , 6(1), 1-28.

Moldovan D. and Harabagiu S. and Pasca M. and Mihalcea R. and Goodrum R. and Girju R. and Rus V. (1999): LASSO: A Tool for Surfing the Answer Net. Proceedings of TREC-8 Conference, NIST, 1999.

Moreale E. and Vargas-Vera M. (2003): Towards a Student Portal. KMI-TR-136, Knowledge Media Institute, The Open University, 2003.

Motta E. (1999): Reusable Components for Knowledge Modelling. IOS Press. Netherlands, 1999.

Noy N. and Musen M. (2000): PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Proc. of the 17th National Conference on Artificial Intelligence (AAAI), 2000.

Plamondon L. and Lapalme G. and Diro R. and Kosseim L (2001): The QUANTUM Question Answering System, Proceedings of TREC-9 Conference, NIST, 2001.

Pulman S. G. (2000): Bidirectional Contextual Resolution, Computational Linguistic Vol 26/4, 497-538, 2000.

Ravichandran D and Hovy E. (2002): Learning Surface Text Patterns for a Question Answering System. In Proceedings of the ACL Conference, 2002.

Resnik P. (1995) Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of IJCAI-95, pages 448-453, Montreal, Canada.

Resnik P. (1998): Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. Journal of Artificial Intelligence research.

Resnik P and Diab M. (2000): Measuring Verb Similarity. Proceedings of the 22nd Annual Meeting of the Cognitive Science Society (COGSCI2000), Philadelphia, USA, 2000.

E. Riloff (1996a): An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. AI Journal, volume 85:101-134, 1996.

E. Riloff (1996b): Automatically Generating Extracting Patterns from Untagged Text. Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)}, pp 1044-1049, 1996.

Stojanovic, N., Maedche, A., Staab, S., Studer, R. and Sure, Y. (2001). SEAL – A Framework for Developing Semantic PortALs, *K-CAP'01*, October 22-23, 2001.

Studer, R., Sure, Y. and Volz, R. (2002). Managing User Focused Access to Distributed Knowledge, *Journal of Universal Computer Science* (*J.UCS*), 8, 6, 662-672, 2002.

Vargas-Vera M. and Motta E. (2004): AQUA - Ontology-based Question Answering System. Accepted to be presented at Third International Mexican Conference on Artificial Intelligence (MICAI-2004), LNAI 2972 Advances in Artificial Intelligence, Springer Verlag, Editor R. Monroy. April 26-30, 2004.

Vargas-Vera M. and Motta E and Domingue J. (2003a): AQUA: An Ontology-Driven Question Answering System. AAAI Spring Symposium, New Directions in Question Answering, Stanford University, March 24-26, 2003.

Vargas-Vera M. and Motta E. and Domingue J. (2003b): An Ontology-Driven Question Answering System (AQUA). KMI-TR-129, Knowledge Media Institute, The Open University, 2003.

Vargas-Vera M. and Celjuska D. (2003c): Ontology-Driven Event Recognition on News Stories. KMI-TR-135, Knowledge Media Institute, The Open University, 2003.

Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A. and Ciravegna, F. (2002). MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. The 13th International Conference on Knowledge Engineering and Management (EKAW 2002), *Lecture Notes in Computer Science 2473*, ed. Gomez-Perez, A., Springer-Verlag, 2002, 379-391, ISBN 3-540-44268-5.

Vargas-Vera M. and Domingue J and Motta E. and Buckingham Shum S. and Lanzoni.M. (2001): Knowledge Extraction by using an Ontology-based Annotation Tool. In proceedings of the Workshop Knowledge Markup & Semantic Annotation, held in conjuction with the First International Conference on Knowledge Capture (K-CAP 2001). Victoria Canada, October 2001, pp 5-12, 2001.

Wu Z. and Palmer M. (1994): Verb semantics and Lexical Selection. 32nd Annual Meetings of the Association for Computational Linguistics, 1994.