AQUA - Ontology-based Question Answering System

Maria Vargas-Vera and Enrico Motta

Knowledge Media Institute (KMi),
The Open University,
Walton Hall, Milton Keynes, MK7 6AA, United Kingdom
{m.vargas-vera, e.motta}@open.ac.uk

Abstract. This paper describes AQUA, an experimental 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 in several parts of the question answering system. The ontology is used in the refinement of the initial query, the reasoning process, and in the novel similarity algorithm. The similarity algorithm, is a key feature of AQUA. It is used to find similarities between relations used in the translated query and relations in the ontological structures.

1 Introduction

The rise in popularity of the web has created a demand for services which help users to find relevant information quickly. One such service is question answering (QA), the technique of providing precise answers to specific questions. Given a question such as "which country had the highest inflation rate in 2002?" a keyword-based search engine 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] [1] 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. 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 in a uniform framework.

The first instantiation of our ontology-driven Question Answering System, AQUA, 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 intranets; for example, giving AQUA an ontology of computer systems might allow it to be used for trouble-shooting or configuration of computer systems.

[1] http://www.ask.com/

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 markup of home pages with semantic content. These annotations can be written in RDF [20, 13] or RDFS [5], 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 annotate 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 (query reformulation), the reasoning process, 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 in the translated query and relations in the ontological structures. The similarities detected then allow the interchange of concepts or relations in the logic 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.

The paper is organized as follows: Section 2 describes the AQUA process model. Section 3 describes the Query Logic Language (QLL) used in the translation of the English written questions. Section 4 presents our query satisfaction-algorithm used in AQUA. Section 5 describes the similarity algorithm embedded in AQUA. Section 6 shows output enhancements of the AQUA system. Section 7 describes a section of related work. Finally, Section 8 gives conclusions and directions for future work.

2 AQUA process model

The AQUA process model generalizes other approaches by providing a 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).

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.

[2] The goal of the Semantic Web is to help users or software agents to organize, locate and process content on the WWW.

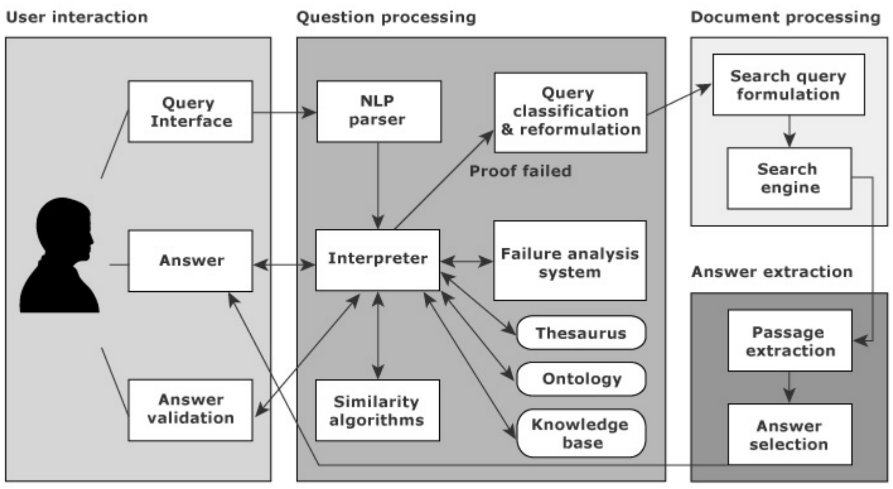

Fig. 1. The AQUA architecture

   - *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 the Resolution algorithm [8].
   - *WordNet/Thesaurus.* AQUA's lexical resource.
   - *Ontology.* This currently contains people, organizations, research areas, projects, publications, technologies and events.
   - *Failure-analysis system.* This analyzes the failure of a 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[3] of the question. Document processing consists of two components:
   – *Search query formulation.* This transforms the original question, Q, using transformation rules into a new question *Q'*. 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 Query Logic Language (QLL)

In this section we present the Query Logic Language (QLL) used within AQUA for the translation from 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 [8, 22]).

Like Prolog or OCML [25], QLL uses unification and resolution [22]. However, in the future we plan to use Contextual Resolution [28]. 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 carried out in order to provide QLL with three-valued logic. Once that QLL become a three-valued logic language then an evaluation of a predicate could produce *yes*, *no* or *unknown* as in Fril [3]. Finally, QLL handles negation as failure but it does not use cuts.

AQUA uses QLL as an inter-media language as is shown in example (section 4).

The 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. The form of the logical predicates introduced by each syntax category is described in detail in [29].

4 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

[3] 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.

following algorithm AQUA uses steps 1-4.1 to evaluate query over the populated AKT reference ontology. [4]. 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 phrases, object and adjectives.
2. Use the ontology to convert from the QLL language 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 formulae 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.

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] [7] 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 find the publications of Yorik Wilks (for example,

[4] 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/)

[5] The pattern-matching algorithm tries to find an exact match with names in the ontology.

[6] http://www.google.com

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

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.

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 to the standard (Prolog-style) expression by using the AKT ontology

$$\exists\, X : \mathit{Domain} \quad \text{technology(X)} \quad \wedge \quad \text{project(akt)} \quad \wedge \quad \text{use(X,akt)}.$$

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.

$$\mathit{commercial\_technology} \quad \subseteq \quad \mathit{technology}$$

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

$$\exists\, X : \mathit{commercial\_technology} \ \text{commercial\_technology(X)} \quad \wedge \quad \text{project(akt)} \quad \wedge \quad \text{use(X,akt)}.$$

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.

5 Concept and relation similarity algorithm

The success of the attempt to satisfy a query depends on the existence of a good mapping between the names of the relations used in the query and the names of the relations used in the knowledge base/ontology. Therefore, we have embedded

[8] http://citeseer.nj.nec.com/cs

a similarity algorithm in AQUA. Our similarity algorithm uses both ontological structures and instances of 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 from either the kind of similarity which can be achieved using only either WordNet or distances to superclasses (Wu et al [31]). In the former approach, WordNet returns all the synsets found, even the ones which are not applicable to the problem being solved. On the other hand, in the latter approach, the idea of a common superclass between concepts is required.

For the sake of space, we present a brief 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 an example can be found in [30].

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 (G2) is examined, including neighborhood classes. Then an intersection[9] (G3) between augmented query G1 and ontology G2 is performed to assess structural similarity. This is done using knowledge from the ontology. It may be the case that, in the intersection G3, several relations include the grounded arguments. In this case, the similarity measure is computed for all the relations (containing elements of the user query) by 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, AQUA presents the user with the synsets obtained from WordNet. From this offered set of synsets, the user can then select the most suitable one.

6 Output enhancements

AQUA not only provides a set of elements which satisfy the query, AQUA enhances its answer using the information from the AKT reference ontology. It provides more information about each element in the set. For instance, if the query is “who works in AKT?” then AQUA 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 [10] in this task of enhancing answers. Magpie is a semantic browser which

[9] Intersect means to find a portion in a ontology G_2 which contains all concepts contained in G_1 (reformulated query) by applying a subsumption relation.

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.

7 Related work

There are many trends in question answering [16, 17, 18, 27, 4, 24, 14, 15, 2], however, we only describe the systems most closely related to the AQUA system philosophy.

MULDER is a web-based QA system [19] 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) [6]. 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 a information retrieval system coupled with an ontology [12]. 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.

8 Conclusions and future work

In this paper we have presented AQUA - a question answering system which amalgamates NLP, Logic, Information Retrieval techniques and Ontologies [10]. AQUA translates English questions into logical queries, expressed in a language, QLL, that are then used to generate of proofs. Currently AQUA is coupled with the AKT reference ontology for the academic domain. In the near future, we plan to couple AQUA with other ontologies from our repertoire of ontologies.

[10] AQUA has been implemented in Sicstus Prolog, C, OCML and PHP.

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 our similarity algorithm embedded in AQUA which 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. Future work, intends to provide AQUA with a library of similarity algorithms.

We will also explore the automatic extraction of inference rules, since knowledge about inference relations between natural language expressions is very important for the question answering problem.

Acknowledgments

This work was funded by the Advanced Knowledge Technologies (AKT). The authors would like to thank Arthur Stutt and Mark Gaved for their invaluable help in reviewing the first draft of this paper.

References

1. Askjeeves: http://askjeeves.com/, 2000.
2. G. Attardi and A. Cisternino and F. Formica and M. Simi and A. Tommasi: Proceedings of TREC-9 Conference, NIST, pp 633-641, 2001.
3. J. F. Baldwin and T.P. Martin and B. W. Pilsworth. Fril - Fuzzy and Evidential Reasoning in Artificial Intelligence. Research Studies Press in 1995.
4. R. D. Burke and K. J. Hammond and V. A. Kulyukin and S. L. Lytinen and N. Tomuro and S. Schoenberg: Questions answering from frequently-asked question files: Experiences with the FAQ Finder System. The University of Chicago, Computer Science Department, 1997, TR-97-05.
5. D. Brickley and R. Guha: Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Web Consortium, 2000. URL:http://www.w3.org/TR/2000/CR-rdf-schema-20000327.
6. E. Breck and D. House and M. Light and I. Mani: Question Answering from Large Document Collections, AAAI Fall Symposium on Question Answering Systems, 1999.
7. F. Ciravegna and A. Dingli, D. Guthrie and Y. Wilks: 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, April 2003.
8. W. F. Clocksin and C. S. Mellish: Programming in Prolog, Springer-Verlag, 1981.
9. A. Doan and J. Madhavan and P. Domingos and A. Halevy: Learning to Map between Ontologies on the Semantic Web. In Proc. of the 11th International World Wide Web Conference (WWW2002), 2002.
10. J. Domingue and M. Dzbor and E. Motta: Semantic Layering with Magpie. Technical Report KMI-TR-125. February, 2003.
11. W. Frakes and R. Baeza-Yates: Information Retrieval: Data Structures & Algorithms, Prentice Hall, 1992.

12. N. Guarino: OntoSeek: Content-Based Acess to the Web, IEEE Intelligent Systems, pp 70-80,1999.
13. P. Hayes: RDF Model Theory, W3C Working Draft, February 2002. URL:http://www.w3.org/TR/rdf-mt/
14. E. Hovy and L. Gerber and U. Hermjakob and M. Junk and C-Y Liu: Question Answering in Webclopedia, Proceedings of TREC-9 Conference, NIST, 2001.
15. E. Hovy and L. Gerber and U. Hermjakob and C-Y Liu and D. Ravichandran: Toward Semantics-Based Answer Pinpointing, Proceedings of DARPA Human Language Technology conference (HLT), 2001.
16. B. Katz: 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.
17. B. Katz and B. Levin: Exploiting Lexical Regularities in Designing Natural Language Systems, MIT Artificial Intelligence Laboratory, 1988, TR 1041.
18. B. Katz: Using English for Indexing and Retrieving, MIT Artificial Intelligence Laboratory, 1988, TR 1096.
19. C. Kwok and O. Etzioni and D. S. Weld: Scaling Question Answering to the Web, World Wide Web, pp 150-161, 2001.
20. O. Lassila and R. Swick: Resource Description Framework (RDF): Model and Syntax Specification. Recommendation. World Wide Web Consortium, 1999. URL: http://www.w3.org/TR/REC-rdf-syntax/.
21. D. Lin and P. Pantel: Discovery of Inference Rules for Question Answering, Journal of Natural Language Engineering, 2001.
22. J. W. Lloyd: Foundations of Logic Programming, Springer-Verlag, 1984.
23. C.D. Manning and H. Schutze: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge Massachusetts. 1999.
24. D. Moldovan and S. Harabagiu and M. Pasca and R. Mihalcea and R. Goodrum and R. Girju and V. Rus: LASSO: A Tool for Surfing the Answer Net. Proceedings of TREC-8 Conference, NIST, 1999.
25. E. Motta: Reusable Components for Knowledge Modelling. IOS Press. Netherlands, 1999.
26. N. Noy and M. Musen: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Proc. of the 17th National Conference on Artificial Intelligence (AAAI), 2000.
27. L. Plamondon and G. Lapalme and R. Diro and L Kosseim: The QUANTUM Question Answering System, Proceedings of TREC-9 Conference, NIST, 2001.
28. S. G. Pulman: Bidirectional Contextual Resolution, Computational Linguistic Vol 26/4, 497-538, 2000.
29. M. Vargas-Vera and E. Motta and J. Domingue: AQUA: An Ontology-Driven Question Answering System, AAAI Symposium on New Directions of Question Answering Stanford University, March 24-26, 2003.
30. M. Vargas-Vera and E. Motta and J. Domingue: AQUA: An Ontology-Driven Question Answering System, KMI-TR-129, KMi, The Open University, 2003.
31. Z. Wu and M. Palmer: Verb semantics and Lexical Selection. 32nd Annual Meetings of the Association for Computational Linguistics. 1994.

This article was processed using the LaTeX macro package with LLNCS style