AQUA: an ontology driven question answering system

Conference or Workshop Item

AQUA: An Ontology-Driven Question Answering System

Maria Vargas-Vera, Enrico Motta and John Domingue

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

Abstract

The use of the web has become popular and also the need of services that could exploit the vast amount of information in it. Therefore, there is a need for automated question answering systems. These kind of systems should allow users to ask questions in everyday language and receive an answer quickly and with a context which allows the user to validate the answer. Current search engines can return ranked list of documents but they do not deliver answers to users.

1. Introduction

In recent years, the use of the web has become popular. Therefore there is a need to provide services which help users to skim all irrelevant information quickly. One of the services is question answering (QA). Question answering is the technique of providing precise answers to specific questions as opposed to document retrieval. Current search engines (based on information retrieval techniques) would not give an answer to questions such as, which country has the highest inflation in 2002? Instead will present web pages from the Financial Times.

A typical example of QA systems, available on the web, is for instance, Jeeves (http://www.ask.com/). Jeeves 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. Then the user selects the most appropriate entry in the list.

Users would usually prefer to be given a specific answer rather than find the answer themselves in a document or make a selection in a list of matching questions. Therefore, an automatic system which could provide with textual answers instead a set of document seems reasonable to be aimed. One method, of course, is to simply aim for the full understanding of the text, however, such in-depth understanding is still out of reach. Instead a solution based in many sorted logics and ontologies might be feasible.

We remark that open-ended question-answering systems that allow users to pose a question of any type without any restrictions, remains beyond the scope of today's text processing systems. We investigate instead a restricted variation of the problem.

Our main goal was to create a question answering system by integrating several technologies such as ontologies, Logic and NLP. Then we had built AQUA where AQUA stands for Automated QUestion Answering System.

AQUA attempts to exploit semantically annotated web pages with the main purpose of answer questions. These annotations could be written in RDF ([Lassila et al. 99]) or RDFS ([Brickley et al. 00]) provide the basic framework for expressing metadata on the web. AQUA uses the semantic annotations to perform inferences and reduce the number of possible answers to a question.

The major contribution of AQUA is the use of an ontology in order to go beyond superficial keyword matching as typical search engines. The AQUA's inference engine operates within the framework of multi-sorted logic, in which every term has a type and every predicate is associated with a domain. Also AQUA has embedded a similarity algorithm which is used in the mapping between names of relations in the knowledge base and names of relations in the ontology (these name of relations are not necessarily the same syntactically).

The paper is organized as follows: Section 2 describes the AQUA's process model. Section 3 presents the architecture of the AQUA system. Section 4 describes the Query Logic language (QLL) used in the translation of the English written questions. Section 5 describes the question classification module. In Section 6 we present the algorithm used in our question/answering system. This algorithm is based in shallow parser, information extraction methodology and inference rules for an specific domain. Section 7 describes the similarity algorithm embedded in AQUA. Section 8 presents related work and finally, Section 9 gives conclusions and directions for future work.

2. AQUA process model

AQUA process model generalize other approaches [Guarino 99; Kwok et al. 01; Breck et al. 99] by providing a uniform framework which integrates logic queries and information retrieval. Within this work we have focused on creating a process model for the AQUA system. In this process model there are three activities which are described as follows:

Question processing. The Question processing is performed in order to understand the question asked by the user. This "understanding" of the question requires several steps such as parse the question, representation of the question and classification of the question on one of the following types: *what*, *who*, *when*, *which why* and *where*.

Document Processing. Document processing relies on the extraction of the focus of the question. Then a set of document is selected and a set of paragraphs are extracted.

Answer processing. Answers are extracted and validated using the information of type of expected answer and then questions are scored.

A detailed architecture of the AQUA system is described in the next section. This architecture has embedded the process model outlined in this section.

3. The AQUA architecture

Figure 1 shows the ideal architecture of our AQUA system. Each module in the architecture is described as follows:

1. Query interface. The user writes a question using the user interface. This query interface is a Google sort type interface. If the user does not obtain a satisfactory answer then he/she could reformulate the query.

2. The **NLP parser** does the segmentation of the sentence into subject, verbs, prepositional phrases, adjectives and objects. The output of this module is the logic representation of the query.

3. WordNet. It is used as dictionary in the AQUA system.

4. Ontology. We use a hand-crafted ontology containing people, projects, publications, technologies and events.

5. Knowledge base. This knowledge base is constructed incrementally and it is domain specific. In our case is knowledge base containing information about our organization such as researchers, projects, publications, technologies and events happening in our institute.

6. Interpreter is the logic interpreter which executes a query using unification and resolution algorithms. It finds a proof of the query against the knowledge base.

7. Failure analysis. This subsystem analyzes the failure of a given question and gives an explanation why the query failed. Then the user could provide new information for the proof. At this point the proof could be re-assume. This process could be repeated several times as is needed.

8. Question classification & reformulation classifies question as belonging to any of the types supported in AQUA (*what*, *who*, *when*, *which*, *why* and *where*).

9. Search query formulation. In this module we transform the original question using transformation rules into a new question **Q'**. At this stage synonymous words are used, punctuation symbols are removed and words are stemmed.

10. Search engine searches in the web for a set of documents which satisfy the query using a selected set of keywords.

11. Answer extraction extracts information from the set of documents that the search engine found satisfying the question **Q'**.

12. Answer selection it has three functionalities. It clusters answers, scores them using the voting model and finally it obtains a final ballot.

We could identify the three processes described in section 2. Steps 1-8 correspond to question processing. Steps 9-10 correspond to the document processing and steps 11-12 correspond to the answer processing.

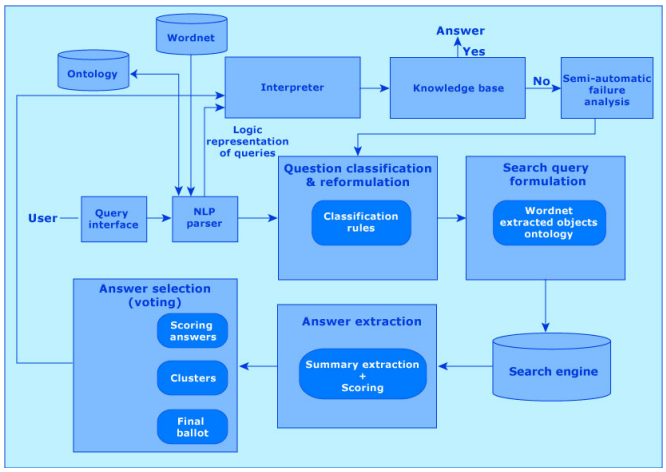

Figure 1. The AQUA architecture

4. Question Logic Language (QLL)

QLL is a query logic language which could be used to express questions about a specific domain. It contains terms in the Prolog sense and these could be defined inductively [Clocksin 81]. The logic language has embedded type definitions for each variable and type for each predicates. QLL is a subset of Prolog. Therefore, the semantics of QLL is weaker than the Prolog semantics. For a full semantics of Prolog refers to [Lloyd 84]. Note that

only variables, atoms, and certain forms of terms might appear as arguments of a QLL predicate.
The translation between a query written in English and a logical form is performed using the following rules of translation. The form of the logical predicates introduced by each syntax category is described as follows:

1. Nouns (without complement) introduce a predicate of arity 1. For example the noun capital could introduce the predicate capital(X) restricting that the type of X is the name of a city.

2. Nouns (with complement) introduce a predicate of arity the number of complements plus one. The pattern is predicate_name(argument1, ..., argumentN). For example, in the question *what is the population of the UK*? the noun population gets translated in the predicate population(uk, X).

3. Qualitative adjectives introduces a predicate of arity 1. The pattern is predicate_name(argument). For example, the adjective *European Community* translates into european(X).

4. Quantitative adjectives introduces a binary predicates. The pattern is the following:
predicate_name(argument_1, argument_2). For example, the question *how big is London*? translates into the following predicate: has-size(london, Y).

5. Prepositions introduce a binary predicate. The pattern is as follows:
name_preposition(argument_1, argument_2). For example, la preposition between gets translated in the predicate between(X,Y).

6. Verbs introduce predicates with one or more arguments. The first argument should be the subject of the verb and the second is the direct object, the third is the indirect object (if any) and complements (if any). For example, *when does lord Putmann visited KMi*? is translated in the following predicate: visited(lord_putmann,kmi).

7. A set of built-in predicates is also available in our QLL language.

5. Questions types

This 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 the segmentation of the sentence. During sentence's segmentation the system finds nouns, verbs, prepositions and adjectives. The category of a desired answers are listed below.

what/which - this kind of questions appear 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 a reason.

where - the category of the answer is location.

The range of answers of a question answering system varies from yes/no answers, true/false questions, and questions which could be answered with 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.

6. AQUA algorithm

The classification information is giving information about the kind of answer that we should expect to achieve as an answer. Therefore we could anticipate the type of answer that the system will produce. For example, if we ask *what is the capital of Mexico*? we know that in the answer is almost certainly the name of a city.
The main algorithm implemented in our AQUA system consists of the following steps:

Algorithm

1. To provide the question Q.
2. To parse the question in its grammatical components.
3. To translate the English question into a logic formulae.
4. To execute the logic formulae against the knowledge base. if succeed then provide an answer and go to step 5.
if not then
To classify question in 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
To transform the query Q into a new query Q'.
To launch a search engine with the new question Q'
To analyze retrieved documents which satisfy the query Q'.
To perform answer extraction
To perform answer selection
5. Stop

7. Algorithm for concept and relation similarity

The success of a query evaluation depends on a good mapping between the names of relations used in the user's query and names of relations used in the knowledge base. AQUA has embedded a similarity algorithm which provide alternative names of relations. Our similarity algorithm is defined using Dice coefficient [Frakes et al. 92] and WordNet. It uses a graph containing a subset of the ontology (with the relevant concepts to the query) and the graph obtained from the query. The output is the degree of similarity between concepts/relations and the alternative relation name. If similarity is below a given threshold then AQUA provides synsets from Wordnet and the user should select one sense of the synsets offered.
The mapping between names in the knowledge base and the query was one of the major problems that we encountered in the design of the AQUA system.

8. Related work

In this section we describe several systems related to the AQUA system.
MULDER is a web QA system related to our work [Kwok et al. 01]. Mulder extracts snippets called summaries and generates a list of candidate answers. However, the system does not have an inference mechanism embedded such as the use of semantic relations defined in an ontology like in the AQUA system.
QANDA 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 sentence) [Breck et al. 99]. QANDA is a Question Answering system which combines knowledge representation, information retrieval and natural language processing. A question is represented as first order logic expression. Also knowledge representation techniques are used to represent questions and concepts discussed in the documents. However, QUANDA does not use ontological relations and domain specific axioms like in AQUA.
Ontoseek is a information retrieval system coupled with an ontology [Guarino 99]. Ontoseek performs retrieval based on content instead of string based retrieval. The target was the information retrieval with the aim of improving recall and precision and the focus was to specific classes of information repositories yellow pages and product catalogues. The Ontoseek system provides interactive assistance on query formulation generalization an specialization. Queries are represented as conceptual graphs then according with the authors "the problem is reduced to ontology driven graph matching where individual node 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 designated user-interface.

9. Conclusions

We had developed a question answering system called AQUA. AQUA uses NLP technology, Logic and an hand-crafted ontology. The main goal of AQUA is to find a textual answer to the question in a short period of time.
The first implementation of AQUA answers questions about KMi domain because we had coupled AQUA with the KMi ontology which consists of people, projects, publications and events. However, in future implementation we plan to provide answers in different domains by coupling AQUA with other ontologies. We had discussed a similarity algorithm embedded in AQUA using and Dice coefficient and WordNet. This algorithm is used by AQUA to ensure that the question does not fail because there is a mismatch between names of relations in the knowledge base and the user query. Finally, as future work we will explore how automatically extract inference rules since knowledge about inference relations between natural language expressions is very important for the question answering problem.

Acknowledgements

This work was funded by the Advanced Knowledge Technologies (AKT), which is sponsored by the UK Engineering and Physical Sciences Research Council.

References

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

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

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

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

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

Katz B. 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.

Kwok C., and Etzioni O., and Weld D.S. Scaling

Question Answering to the Web, World Wide Web, pp 150-161, 2001.
Lassila O., and Swick R.Resource Description Framework (RDF): Model and Syntax Specification. World Wide Web Consortium, 1999. URL:http://www.w3.org/TR/REC-rdf-syntax/.
Lloyd J. W. Foundations of Logic Programming, Springer-Verlag, 1984.