International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:6, 2009

An Ontology Based Question Answering System on Software Test Document Domain

Meltem Serhatli, Ferda N. Alpaslan

***Abstract*—**Processing the data by computers and performing reasoning tasks is an important aim in Computer Science. Semantic Web is one step towards it. The use of ontologies to enhance the information by semantically is the current trend. Huge amount of domain specific, unstructured on-line data needs to be expressed in machine understandable and semantically searchable format. Currently users are often forced to search manually in the results returned by the keyword-based search services. They also want to use their native languages to express what they search. In this paper, an ontology-based automated question answering system on software test documents domain is presented. The system allows users to enter a question about the domain by means of natural language and returns exact answer of the questions. Conversion of the natural language question into the ontology based query is the challenging part of the system. To be able to achieve this, a new algorithm regarding free text to ontology based search engine query conversion is proposed. The algorithm is based on investigation of suitable question type and parsing the words of the question sentence.

***Keywords*—**Description Logics, ontology, question answering, reasoning.

I. Introduction

IRRESPECTIVE of the domain, the main aim of a Question Answering system is getting a question from the user, comprehending it, searching the answer in an efficient way and presenting the answers to the user. Many methods have been devised for this purpose [1], [2], [3], [4], [5], [6], [7], [9]. In this paper, an automated question answering system on software test document domain is presented. This basic idea is using an ontology for representing the knowledge and developing the knowledge base. Although the ultimate aim of question answering is finding the exact answer to any question in any context, in today's world of automated content processing, this is inherently a hard task because without a restriction imposed either on the question type or on the user's vocabulary, the question answering process gets a big hit even at the question interpretation phase. This is why, most of the efforts are focused on answering "factoid style" questions, since it is much more efficient to use through text processing algorithms based on pattern extraction or information retrieval techniques.

Speaking in Description Logics terms [8], the ontology has provided the basic classes, their properties and their relations between themselves as TBoxes. Moreover, the individuals, which are initializations of the classes aforementioned, are represented as ABoxes. With this approach, the question answering mechanism simply becomes as a set of reasoning tasks over the ontology. The aim of this work is constructing a question answering system using document ontology. The ontology to be used contains both the information semantics in the shape of TBoxes and the answers to the questions as ABoxes. The question answering system provides an authoring environment which facilitates content sharing by automatically tagging content with semantic metadata and by using open standards to store it in networked repositories supporting symbolic and similarity-based indexing and search capabilities for all content types.

This work will present a free text to ontology based search engine query conversion algorithm. The idea is to control the matching of the words in the question sentence to the classes and attributes, if the parsing with words cannot be done then to find classes and attributes by looking at the rules. If the related classes and attributes cannot be found, then to look for the synonyms of the words in WordNet.

This paper is organized as follows: After briefly giving the motivation and related work in section II; section III describes the implementation of the proposed algorithm; section IV presents some results of the new algorithm on a real life case problem from software test document domain; section V draws the conclusions and future work.

II. Motivation and Related Work

Although it is met with difficulties to find applications similar to the proposed one, it is encountered several works dealing with querying ontologies in NL. The most closely related approach is Kaufmann et al. [9]. They present a natural language interface to semantic web querying. The interface allows formulating queries in Attempto Controlled English (ACE), a subset of natural English. Each ACE query is translated into a discourse representation structure – a variant of the language of first-order logic – that is then translated into an N3-based semantic web querying language using an ontology-based rewriting framework. On the other hand, they presents some limitations of their approach. First, the use of a

M. Serhatli is with the Vodafone Telecommunication A.S., Istanbul, 34398 Turkey (phone: 212-367-0509; e-mail: meltem.serhatli@vodafone.com). She is also with the Department of Computer Engineering, Middle East Technical University, Ankara, 06531 Turkey (e-mail: e1305416@ceng.metu.edu.tr).

F. N. Alpaslan is with the Department of Computer Engineering, Middle East Technical University, Ankara, 06531 Turkey (fax: 312-210-5544; e-mail: alpaslan@ceng. metu.edu.tr).

controlled language imposes a cost on the user since the language has to be learned. Users might be discouraged from employing a language they have to learn, but experience with ACE has shown that learning a controlled language to phrase statements and queries is much easier than learning logic, and takes only a couple of days for the basics and two weeks for full proficiency, which is beyond what users need to write queries. Second, their current prototype requires some manual adaptation of the rewrite rules when using it with a new ontology or new knowledge base. The approach differs from ours in that usage of a controlled language requires a learning phase whereas there in no necessity like this in our approach. They reformulate the user queries in ACE whereas the proposed system does not reformulate the queries to any subset of natural english. Another project which is found similar to the proposed one is a simple application of Description Logics in [2]. They present a system which allows students to enter a freely formulated question about computer history. The system returns a very short commented list of multimedia clips where the users find the answer to their questions. The most important part is to find the semantically suitable clip(s). The system uses a knowledge base with 300 multimedia clips that cover the principal events in computer history. The user enters a question in natural language and the system returns a list of suitable clips as an answer. The semantic search engine gets a NL question from the user and maps it to a general assertion. To make it possible, it firstly looks for semantically important words and translates them into RDF. A specific domain dictionaryis used to retrieve the semantics for every word in the sentence. Semantically unnecessary words like {what, did} or too general words like {operating, system} are left out of account. After that, this transformed question is mapped to a general assertion. The set of general assertions is given to the system, and generally contains only few elements. In [2], the question is being mapped to the general assertion. Based on that interpretation, an RDQL query is generated and started against the knowledge base. They achieved a interpretation of a user question in [2] in two steps: the mapping of concepts over the TBox, and the transformation of the user question into an ABox query. For the mapping of concepts, the authors modified a matching algorithm in order to use it in the system. Firstly, they improved the reasoning mechanism in order to perform a query over a non-empty ABox. Secondly, because of dealing with multimedia clips where a textual content is not available, they considered metadata rather than the documents content. On the other hand, one of the principal problems of the solution presented in [2] is the matching algorithm which was created for being used with WordNet as knowledge source. The authors thought that a large-scale dictionary like WordNet was not the best potential solution for a domain ontology about computer history. Therefore, their information retrieval system required setting the different interpretations in a context to find the best match. In addition, large-scale dictionaries often lack specific domain expressions. Because of these reasons, they proposed either to use an existing domain specific dictionary or to create a dictionary of its own. Proposed approach differs in such a way that it created its own document ontology as a domain specific dictionary and its matching algorithm differs from theirs.

The other project which is found relevant to the proposed one is named Aqualog [1]. They introduces a portable question answering system with the techniques for making sense of NL queries and mapping them to semantic markup in [1]. These techniques are listed as follows:

- It makes use of the GATE NLP platform in linguistic component,
- String metrics algorithms ,
- WordNet,
- Novel ontology-based similarity services for relations and classes to make sense of user queries with respect to the target knowledge base.

They defined their system as a waterfall model. In this model, first of all a natural language query gets translated into a set of intermediate, triple-based representations, after that query triples and finally these are translated into ontology-compatible triples. Because of the two basic reasons, they preferred to use a triple-based data model. First reason is the possibility of representing most queries as triples. The other one is RDF-based knowledge representation formalisms for the semantic web, such as RDF itself or OWL also subscribe to this binary relational model and express statements as <subject, predicate, object>. In contrast to our approach, AquaLog combines a learning component, which ensures that the performance of the system improves over time.

III. THE PROPOSED APPROACH

A. System Design

For supporting concurrent users, increasing accessibility and scalability, the system is designed in a web based fashion. Model-View-Controller design pattern is applied in this work. The interface of the system is completely separated from the business logic part as a web application. The most difficult part of an automated question answering application can be identified as comprehending the question to be asked. To get both what has been asked and where to be searched, a common ground must be set for representing the semantic layer of the document domain. Description Logics has been used for that purposes. The document ontology used in this work is in OWL-DL language. Moreover, OWL is a widely accepted ontology language. Reasoning and querying operations are handled by Pellet reasoner which provides complete and efficient algorithms to answer queries. It complies with OWL formal semantics. It, first, loads the ontology and accepts requests from, possibly many, different interfaces. Web applications that fulfill the DIG Interface can easily communicate with the Pellet Server. The DIG Interface is the common interface for Description Logics applications designed by DL Implementation Group in XML. For our purpose, the DIG interface is used to send the queries to the

Pellet DIG Server and get the answers back. The main flow of action is initiated by the user request, which is posted from the question page. SearchServlet gets that request and the system accesses to the Pellet reasoner and obtains the answer from there. Thus, in this work, it is tried to find the answers of questions about a software test document domain, which can be constructed in a limited way. The main types of the questions were who, what, when, which and how many. The answer extraction was held by means of an ontology designed for the domain under concern. In Fig. 1, it is shown an overview of the system.

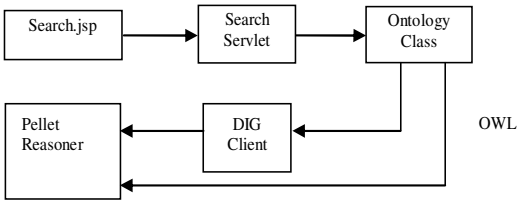

Fig. 1 Overview of the system

B. Implementation

In the system, the action is initiated by entering the question and pushing the "Search" button. In Fig. 2, a screenshot shows an example of a question and the returned answers. After loading and reading the ontology for preparing the model, the process of parsing the sentence is started. As a first step of this process, the query sentence is formed by an object derived from *Sentence(question, true)* class. By using this object, the type of the sentence is investigated and the suitable question type is tried to found for the question sentence. As a second step, the words of the sentence are kept for parsing (except "the", "a", "an", "of", "is", "was", "has", "who", "what", "when", "which", "how", "many ", "much", "why"). While parsing the sentence, the strategies given below are taken into consideration :

- Matching of the words to the classes and attributes being controlled
- If the parsing with words cannot be done, it is tried to find classes and attributes by looking at the rules.
- If the related classes and attributes cannot be found, it looks for the synonyms of the words in WordNet.

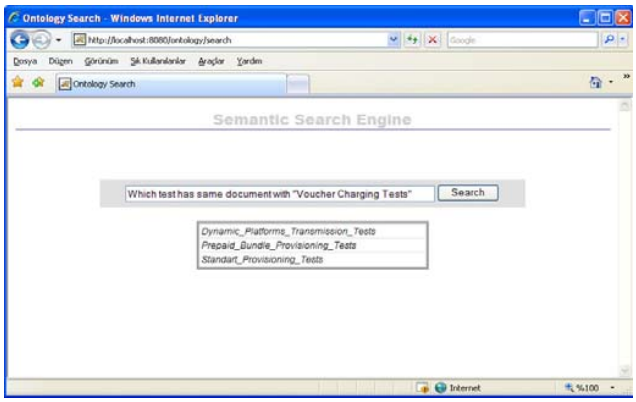

Fig. 2 Image of the screen where the users enter the question and get the answers

After finishing the parsing, the query is built. While forming the query, question sentence types and auxiliary verbs are taken into consideration. Although individual names are not entered, they are taken into consideration with the closest individual names. For example, *"Telecomm_Journal" released* is an example of question sentence with missing words. To be able to find the closest individual name, *fuzzy string matching* method is used Lastly, the query is sent to the reasoner by means of DIG interface. After getting and processing the query by the reasoner, it is sent to the web page over the SearchServlet. Finally, the answer is displayed to the users.

SQWRL is used for the querying of OWL ontologies. It is a SWRL-based query language. SQWRL queries are not independent of SWRL rules in an ontology. They can function in conjunction with those rules. SQWRL queries can thus be used to retrieve knowledge inferred by SWRL rules. In Fig. 3, a screenshot shows the defined rules to be used in our Q&A System.

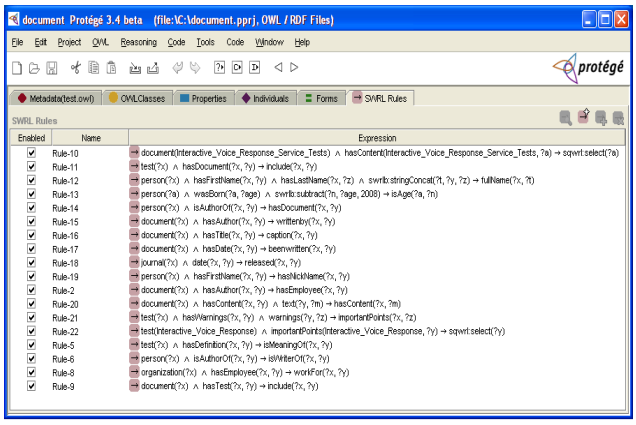

Fig. 3 The figure shows the rules defined in Protégé editor

Here is an example of SQWRL query which finds *"all the other tests belong to Voucher Charging Tests document"*. As test is included in the Document ontology-model as the hasDocument property, it can be translated natural language

query into the SQWRL query as shown in below example

query : test<Voucher_Charging_Tests> ?
hasDocument<Voucher_Charging_Tests,?a> ?
test<?b> ? hasDocument<?b,?a> ? sqwrl:select<?b>

It is asked members of the software test team of a company to enter the queries, which search for test documents that would be of interest to them. Fig. 4 shows a selection of these queries.

What is the important points that should be considered in "Interactive Voice Response"
what is the full name of persons which worked in "organization1"
Which documents are written by authors worked in "organization1"
who is the generator of "Interactive Voice Response"
how many test is included in "INOX Provisioning Tests Document"
what is the total of documents which are written by "Author A"
what is the full name of persons which worked in "organization1"

Fig. 4 A selection of real-world NL queries from which are generated correct SQWRL queries

As it is mentioned before, *fuzzy string matching* method is used as a part of the proposed approach. It helps to find the closest individual names which are missing in the queries entered by the user. Fig. 5 shows a selection of these queries.

what title "message Service"
which document min num test
what total written "Author A"
who writer "Short Message Service"
caption "Short Message Service"
which test same document "Voucher Charging Tests"
results "sms test case"
full name work in "organization1"
generator of "INOX Provisioning Tests"
which test equal type "Message Service Tests"
"Telecomm Journal" released
"Author A" born

Fig. 5 A selection of real-world NL queries with missing words in the question sentence

IV. Results of the Proposed Algorithm

In order to validate the proposed approach, the approach has been tested with a real-life project in software test document domain. As the first step of the experimentation phase, a comparison is made between the expected answer of the questions and the found answer of the questions for each dataset used in the testing session. For getting the success rate, a set of formulas is used as follows:

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{numbers of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$\text{false positive rate} = \frac{\text{number of false positives}}{\text{total number of negative instances}}$$

$$\text{false negative rate} = \frac{\text{number of false negatives}}{\text{total number of positive instances}}$$

Therefore, number of true positives, true negatives, false positives and false negatives are counted for each data set used in testing session .

According to the counted numbers and the above formulas; accuracy of this approach becomes 91%, precision becomes 35% and recall becomes 84% for this approach. By using the last two formulas, false positive rate becomes 9% and false negative rate becomes 7%.

V. Conclusion and Future Work

This paper described a Q&A system, based on an ontology on software test document domain. It explained how the ontology is constructed and a query in natural language is received and transformed into an expression that can be asserted to a Description Logics reasoner. The aim for implementing this system was to provide exact answers to the questions asked by the newly coming members of a software test team. To summarize, although the results obtained are accurate, the work presented in this paper can be extended in several directions: The type of the asked questions can be increased while tailoring the document domain in a way that it includes many more different classes and properties in different documents. In this process, the known document taxonomies have to be better exploited in order to capture real world semantics. Performance issues can be enhanced. For each new question, the reasoner is loading the ontology into the memory. This must be avoided at all costs. There might be some improvements to the term expansion and query relaxation strategies for getting more precise answers. Moreover, much trickier questions that require complex automated reasoning processes can also be handled by the system.

Acknowledgment

This work was fully supported by the Department of Computer Enginering in Middle East Technical University and partially supported by the Vodafone Telecommunication A.S in Turkey. M. Serhatli and F.N. Alpaslan would like to thank

those members of the software test team who took part in the phase of entering the queries to the system.

REFERENCES

[1] V. Lopez, M. Pasin, E. Motta, AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In Proceedings European Semantic Web Conference, Crete (2005)
[2] S. Linckels, C. Meinel, In Proceedings of the IADIS International Conference of Applied Computing (IADIS AC2005), Vol. II, pp. 306-311, Lisbon, Portugal (2005)
[3] Z. Li, K. Ramani, Ontology-based design information extraction and retrieval. Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 21(2), pp. 137–154, 2007.
[4] L. Cinque, A. Malizia, R. Navigli, "OntoDoc: An Ontology-Based Query System for Digital Libraries," icpr,pp.671-674, 17th International Conference on Pattern Recognition (ICPR'04) - Volume 2, 2004
[5] J. Guan, X. Zhang, J. Deng, Y. Qu, "An Ontology-Driven Information Retrieval Mechanism for Semantic Information Portals," skg,pp.63, First International Conference on Semantics, Knowledge and Grid (SKG'05), 2005
[6] U. Hermjakob, Parsing and Question Classification for Question Answering. In Proceedings of the ACL Workshop on Open-Domain Question Answering (2001)
[7] S. Narayanan, S. Harabagiu, "Question Answering based on Semantic Structures". The 20th International Conference on Computational Linguistics (COLING 2004), Geneva (August 2004)
[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Schneider, P, The Description Logic Handbook. Cambridge University Press, 2003
[9] A. Bernstein, E. Kaufmann, A. Göhring, C. Kiefer, “Querying Ontologies:
A Controlled English Interface for End-users”, The Semantic Web – ISWC 2005, Volume 3729/2005, pp. 112-126, 2005