IJCLA VOL. 1, NO. 1-2, JAN-DEC 2010, PP. 233-247
RECEIVED 03/12/09 ACCEPTED 16/01/10 FINAL 10/03/10

Interactive QA using the QALL-ME Framework

IUSTIN DORNESCU AND CONSTANTIN ORĂSAN

University of Wolverhampton, UK

ABSTRACT

One of the main concerns when deploying a real-world QA system is user satisfaction. Despite the relevance of criteria such as usability and utility, mainstream research usually overlooks them due to their inherent subjective, user-centric nature and the difficulty of the evaluation involved. This problem is particularly important in the case of real-world QA systems where a "0 results found" answer is not very useful. This paper presents how interaction can be embedded into the QALL-ME framework, an open-source framework for implementing closed-domain QA systems. The changes necessary to improve the framework are described, and an evaluation of the feedback returned to the user for questions that have no answer is performed.

1 INTRODUCTION

The need to access information and find answers to questions in vast collections of documents led to the emergence of the field of Question Answering (QA). Despite extensive research in this field, the accuracy of open domain question answering system, i.e., systems that can answer any question from any collection of documents, is still rather modest. For this reason, real-world question answering systems are usually closed domain, which means that they are built for very specific domains and exploit domain knowledge to answer questions [1].

One of the main concerns when deploying a real-world QA system is user satisfaction. Despite the relevance of criteria such as usability and utility, mainstream research usually overlooks them due to their inherent subjective, user-centric nature and the difficulty of the evaluation involved. QA benchmarks and evaluation fora (such as TREC[1] or CLEF[2]) usually focus on achieving highly accurate and robust systems, and quantify their performance in terms of precision and recall. We argue that a successful deployment of QA systems should not solely rely on the correctness of the answers, but also on how they interact with users and satisfy their needs. This was acknowledged by the increasing interest in interactive question answering [2–4].

The QALL-ME project [5] has developed a framework for implementing question answering systems for restricted domains. The first implementation of this framework was for the domain of tourism, but it is not bound in any particular way to this domain as demonstrated by its adaptation to the domain of bibliographical information [6]. Despite its flexibility, the framework lacks built-in support for user interaction. This paper demonstrates how it is possible to embed interactivity in the existing framework. The remainder of the paper is structured as follows: Section 2 briefly presents the overall QALL-ME project and framework. The technique used to embed interaction in the framework is presented in Section 3. An evaluation of user satisfaction of the interactivity is presented in Section 4, and the paper finishes with a discussion and conclusions.

2 THE QALL-ME PROJECT AND FRAMEWORK

QALL-ME (Question Answering Learning technologies in a multiLingual and Multimodal Environment) is an EU-funded project with the objective of developing a shared infrastructure for multilingual and multimodal open domain Question Answering.[3] It allows users to express their information needs in the form of multilingual natural language questions using mobile phones and returns a list of ranked specific answers rather than whole web pages.

Language variability, one of the main difficulties of dealing with natural language, is addressed in QALL-ME by reformulating it as a textual entailment recognition problem. In textual entailment, a text (T) is said to entail a hypothesis (H), if the meaning of H can be derived from

[1] http://trec.nist.gov/
[2] http://www.clef-campaign.org/
[3] More information about the QALL-ME project can be found at http://qallme.fbk.eu

the meaning of T. To this end, each question is treated as the text and the hypothesis is a procedure for answering the question [7].

The QALL-ME framework is an architecture for multilingual question answering (QA) systems that can answer questions from structured data sources for freely specifiable domains.[4] In a closed-domain QA system, a question can be viewed as a composition of constraints regarding instances, types and the relations between them. The QALL-ME Framework does the following:

– reliably identifies constraints with respect to a domain modelled by an ontology
– creates the SPARQL query corresponding to the question interpretation
– retrieves the results from a data repository

The first implementation of the framework was for the domain of tourism which will be used for the examples in this paper. The QALL-ME framework is based on a Service Oriented Architecture (SOA) which, for this domain, is realised using the following web services:

1. **Context providers**: used to anchor questions in space and time in this way enabling answers to temporally and spatially restricted questions
2. **Annotators**: identify different types of entities in the input question. Currently three types of annotators are available:
   – named entity annotators which identify names of cinemas, movies, persons, etc.
   – term annotators which identify hotel facilities, movie genres and other domain-specific terminology
   – temporal annotators that are used to recognise and normalise temporal expressions in user questions
3. **Entailment engine**: used to overcome the problem of user question variability and determine whether a user question entails a retrieval procedure associated with predefined question patterns.
4. **Query generator**: relies on an entailment engine to generate a query that can be used to extract the answer to a question from a database. For the tourism demonstrator the output of this web service is a SPARQL query.[5]

[4] http://qallme.sourceforge.net/

[5] SPARQL is a query language defined by the W3C RDF Data Access Working Group which can be used for accessing RDF graphs. It is defined

5. **Answer pool**: retrieves the answers from a database using the query produced by the query generator. In the case of the tourism demonstrator, the answers are extracted from RDF encoded data using SPARQL queries.

The answers, encoded as an RDF graph, are passed to a presentation module which is domain dependent and is not defined in any way by the framework. The interaction between services and the cross-lingual capabilities of the system are realised with the help of a domain ontology, which in the case of the first prototype is described in [8]. The ontology is also used to determine the format in which data is stored and to construct SPARQL queries that are used to access the RDF graph.

The services described above are called by a QPlanner that decides which one should be called depending on the setting: monolingual or cross-lingual (for more details see [5]). One of the drawbacks of the existing QPlanner is that it is only feed-forward, meaning that if a question does not have an answer there is no way to inform the user and allow any form of interaction. In the context of QALL-ME, [9] proposed a way to interact with the user, but the approach does not use the existing framework and requires a completely new implementation. The next section discusses how it is possible to integrate interaction with minimum changes to the existing services.

3 PROVIDING SUPPORT FOR INTERACTION WITH THE USER

Most QA systems have a very basic level of interactivity consisting of independent (question, response) pairs. This type of interaction can quickly become frustrating for the user unless the accuracy of the system is very high. Unlike their open-domain counterparts, closed-domain systems embed enough knowledge to successfully address most correct questions relevant to the domain. However, misunderstandings can occur and the system should provide feedback regarding its ‘understanding’ of the question, thereby helping the user quickly identify the source of misunderstanding. If the interaction medium is extended to accommodate the user’s feedback using either natural language templates (e.g. *No, I did not ask about ... I wanted to know ...*) or via an interactive user interface (Web, mobile clients), then a feedback loop is created which promotes

in terms of the W3C’s RDF data model and will work for any data source that can be mapped into RDF. More information can be found at: http://www.w3.org/2001/sw/DataAccess/

a more natural interaction with the user and continuously improves responses from the system.

The treatment of questions that yield no answer should be an important part of any real-world QA system. The first step in clarifying why there are *0 results found* consists of providing feedback regarding the interpretation of the question, as mentioned above. When the cause is not a misunderstanding, but an overly-specific question, the system should explain why there is no answer and suggest possible changes to the original question, encouraging the user to pose additional questions, e.g. by suggesting more general questions with relaxed constraints, or suggesting alternative constraints which do yield relevant answers. For example, if the user asks *Where can I see Matrix in Wolverhampton tonight?* and there is no such screening, it is not useful just to display *No results*. Users may find an answer such as *There is no screening of the movie Matrix in Wolverhampton tonight. Do you want to find out "What movies can I see in Wolverhampton tonight?"* more appropriate. This gives them the opportunity to either accept the suggestion and be presented with the information, or pose a different question initiating a new cycle.

In QALL-ME, processing a question ends once the data is retrieved from the database. Results are presented to the user by a presentation module which is specialised for a particular interaction medium. To enable the behaviour suggested above, the presentation module needs more than just a SPARQL query and the actual results. This is mainly because the SPARQL query only encodes the semantics of the question implicitly, while the presentation module needs explicit meta-data about the system's understanding/interpretation to suggest viable alternatives to the user. In the current QALL-ME architecture, the actual interpretation occurs during the query generation and the entailment engine stages. For this reason, the semantic information is not directly accessible to the QPlanner. The solution is therefore to augment the output returned by the Query Generator (i.e. the SPARQL query) with meta-data which makes the question interpretation explicit, e.g. in terms of EAT and the constraints identified in the question.

In this section, we show how interaction with the user can be achieved without changing the architecture of the QALL-ME Framework. The proposed mechanism consists of two main parts: 1) injecting meta-data explicitly encoding the system's understanding of the question with respect to the underlying domain ontology, and 2) formulating

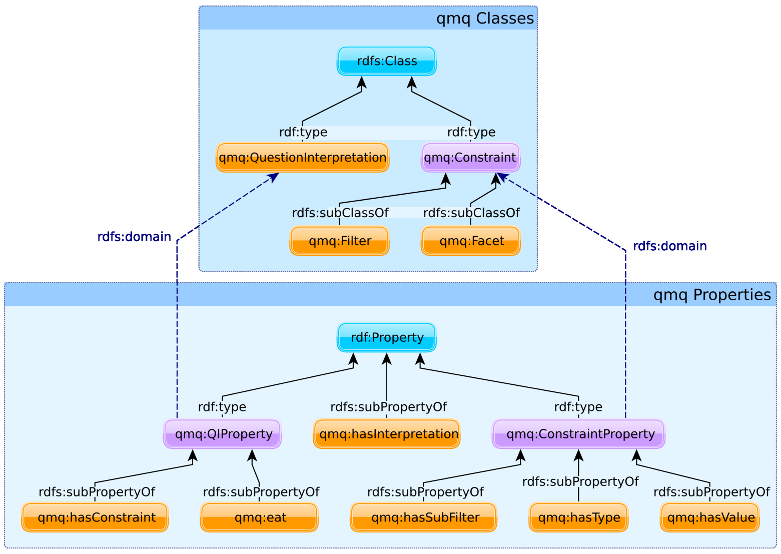

Fig. 1. The qmq terminology

informative answers based on the results given and the question interpretation meta-data. Each of these is discussed below.

3.1 *The qmq mechanism (injecting meta-data)*

In order to provide maximum flexibility, we chose to encode the necessary meta-data using an RDFS terminology, and leave implementation details and extensions to be tailored for each actual application. The terminology contains only the basic concepts: expected answer type, question constraint and question interpretation. The mechanism does not require any changes to the current Web Services specification of the QALL-ME framework, being compatible with the current prototype implementations. An added bonus is that the RDFS terminology used for representing the semantic interpretation is extensible, in line with the generic character of the QALL-ME framework, allowing other types of semantic interpretations to be added in future, without breaking existing applications. Figure 1 presents the qmq terminology.

As mentioned above, the output of the Question Generation service is a SPARQL query that can extract the answer to a question. Schematically this query is represented by the following template:

```
[[prefix declarations]]
CONSTRUCT{
  [[triples containing the results: qmo]]
  [[additional information: qmo]]
  [[answer meta-data: qma]]
}WHERE{
  [[triples for identifying solutions]]
  [[filters for grounding the constraints]]
}
```

In order to accommodate the new features, the base SPARQL template is changed by injecting additional information:

```
[[prefix declarations]]
CONSTRUCT{
  [[triples containing the results: qmo]]
  [[additional information: qmo]]
  [[answer meta-data: qma]]
  [[interpretation meta-data: qmq]]
}WHERE{ OPTIONAL {
  [[triples for identifying solutions]]
  [[filters for grounding the constraints]]
}}
```

Adding the encompassing OPTIONAL keyword ensures that the meta-data triples from the CONSTRUCT part are generated when querying the data-store, even if no actual solution is found. This means that the presentation module can use this extra information to generate informative answers.

3.2 *Generating Feedback: question interpretation*

Providing feedback regarding the system’s understanding helps the user to easily identify misinterpretations, allowing them to rephrase the question in a way that would eliminate the cause of ambiguity or error. The presence of the question interpretation meta-data in the retrieved RDF graph (even in the absence of actual results), enables the presentation module to describe the systems understanding of the question to the user. Instead of saying: *0 results found*, the presentation module can use the meta-data to say: *No action movies are shown between 12 and 18 October in Wolverhampton, West Midlands*.

For the above example, the triples added to the CONSTRUCT section to enable feedback generation are:

```
prefuri:qi rdf:type qmq:QuestionInterpretation;
       qmq:hasInterpretation "[GENRE] movies
               that will be showed
               during TIME] in [DESTINATION]"@en;
       qmq:eat qmo:Movie.
```

The `prefuri` prefix can be a standard prefix or a dereferenceable URI associated with the user session, enabling real dialogue interaction. The qmq:hasInterpretation property contains a natural language explanation, which is a form of textual feedback to be shown to the user, but it can also be the URI of a resource from a custom repository encoding more complex information (e.g. HTML generation templates).

The actual implementation is application dependent. The content generation templates can be part of a resource which uses/extends the qmq terminology to accommodate different presentation media, multilinguality, dialogue management, etc. We chose the RDFS semantics for specifying the terminology in order to maintain flexibility. An actual implementation could use a richer representation schema if necessary.

3.3 *Informativeness: Filters*

To find information quickly, users need a certain level of familiarity with the system such as how to best pose questions, the kind of requests the system can address and the data that the system can access. In order to facilitate and create a more natural interaction, the system should go beyond displaying lists of results by generating informative answers. A straight-forward way to do this is by extending the meta-data describing the constraints.

In cases where a question does not yield any results, the system should be able to suggest ways in which the constraints can be successfully relaxed, to find some results. The qmq:Filter instances should therefore mark the value that enforces the constraint. In the following listing we give an example of such meta-data:

```
prefuri:c1 rdf:type qmq:Filter;
       qmq:hasInterpretation "Movies
                    in [DESTINATION]";
       qmq:hasType qmo:Destination;
       qmq:hasProperty qmo:name;
       qmq:hasValue '''FILTER (?destName =
                    "[DESTINATION]").'''.

prefuri:c2 rdf:type qmq:Filter;
       qmq:hasInterpretation "Movies during
                    [TIMEX]";
       qmq:hasType qmo:DatePeriod;
       qmq:hasProperty qmo:startDate;
       qmq:hasValue '''[TIMEX2]'''.
```

Each of the filters indicates the SPARQL filter clause enforcing it via the property qmq:hasValue. Using this meta-data, the presentation module can remove the filter from the SPARQL and pre-emptively check if ignoring this constraint yields any/additional results. This is particularly useful in cases where questions yield no answers: the system can suggest alternatives by removing the filtering clause from the SPARQL, querying the data-store again and identifying alternatives. For example, by relaxing the spatial constraint `c1`, the system can find which Destinations (qmq:hasType) actually yield results, and extract their names (qmq:hasProperty).

The way such information is displayed depends on the application and the medium used. A textual answer could be: *No movies found during 'this week' within 'Wolverhampton'. Try another DatePeriod(5 movies) or another Destination(12 movies)*. On a mobile device, a map can be displayed with the number of movies available for every Destination, and on the Web the interface can be much richer: a full list of answers with reviews, ratings, times. The system can have several strategies for generating answers and only display the highest ranked ones based on their informativeness.

4 EVALUATION OF THE ANSWER FEEDBACK COMPONENT

As explained in the previous section, one way to deal with questions with no answers is to relax or remove their constraints. However, there are various ways in which these constraints can be changed. In this section we present an evaluation where a set of 6 questions, 3 from the domain of movies/cinema and 3 from the domain accommodation, were shown to users telling them they have no answers and showing them alternative questions that can be generated by the system as part of the response. Users were asked to rate each alternative question on a scale from 1 to 4, 1 indicating a very bad alternative, and 4 corresponding to an excellent alternative. 31 participants were involved in the experiment.

The questions considered for this experiment contained constraints about time (*this weekend*, *tonight*), location (*Wolverhampton*), movie name (*Matrix*), facilities (e.g. *disabled access*), movie genre (*horror movie*), hotel rating and room price. We selected these constraints as they are important in user questions and cover a wide range of concepts from the ontology. The following list of questions was used:

1. Where can I see Matrix in Wolverhampton tonight?
2. Where can I see Matrix in Wolverhampton this weekend?
3. What is the name of a hotel in Wolverhampton with disabled access?
4. What horror movie can I see in Birmingham on Friday night?
5. Where can I find a four star hotel in Wolverhampton?
6. What is the name of a hotel in Wolverhampton where single rooms cost less than £57?

For each of these questions between 4 and 6 alternative questions were produced, in addition to a reply indicating that there are no answers. For example, the following alternatives were proposed for the first question:

1. There is no screening of the movie Matrix in Wolverhampton tonight.
2. Do you want to find out **What movies** can I see in Wolverhampton tonight?
3. Do you want to find out **Where** can I see Matrix tonight?
4. Do you want to find out **When** can I see Matrix in Wolverhampton?
5. Do you want to know Where can I see Matrix in Wolverhampton **tomorrow**?
6. Do you want to know Where can I see Matrix in Wolverhampton **tomorrow evening**?
7. Do you want to find out Where can I see Matrix in **Birmingham** tonight?

A score was calculated for each alternative option for a question as the average rating assigned to it by the users. Tables 1 and 2 present these

Table 1. Results for the question in the movie/cinema domain

		1	2	3	4	5	6	7
Q1	avg.score	2.61	3.42	3.35	3.23	2.45	*2.16*	2.32
	rank	2	1	1	1	2	*3*	2,3
Q2	avg.score	2.65	3.29	3.13	3.06	2.55	2.35	
	rank	2	1	1	1	2	2	
Q4	avg.score	2.61	2.90	2.45	3.13	*1.65*	2.39	
	rank	2	1,2	2	1	*3*	2	

scores. Paired T-test was used to calculate whether there is a statistically significant difference between the answers.

In our domain, spatially and temporally restricted questions are very common, therefore it is important to suggest follow-up questions that are likely to be useful to users. In the first two questions we investigate if the granularity of the temporal constraint from the question (*tonight* vs. *this weekend*) has an impact on the usefulness of alternative time spans. Table 1 presents the average score for each option. The difference between options in the same rank is not statistically relevant. Suggesting a particular alternative value for the temporal constraint (e.g. *tomorrow* instead of *tonight*) was considered less useful than showing all the available alternatives (options 5 and 6 vs. 4 in both questions). This is also true for the spatial constraint (option 7 vs. 3 in Q1). In both questions, the three options that inform the user of all the available alternatives (options 2, 3 and 4) were consistently rated the most useful. This suggests that the users want to know what their options are before committing to a decision.

In question 4 there are three constrains: temporal, spatial and film genre. The results show that option 4 (which movies are available, regardless of the genre) is very useful, while option 5 (which romantic comedies are available) is the least useful and the other options are not statistically distinguishable from the reference option. The results are consistent with our findings so far: ignoring the genre constraint and listing the available movies (option 4) is more useful than pre-emptively modifying the initial question with a viable alternative (options 2, 3, 5 and 6). Users found *romantic comedies* (option 5) a much less desirable alternative to *horror* movies than *action thrillers* (option 6), suggesting

Table 2. Results for the question in the accommodation domain

		1	2	3	4	5	6
Q3	avg.score	2.45	*1.26*	2.23	3.13	*1.55*	3.58
	rank	3	*5*	3	2	*4*	1
Q5	avg.score	2.61	2.84	3.19	3.32	2.29	
	rank	2,3	2	1	1	3	
Q6	avg.score	2.68	*2.03*	3.03	2.94	3.35	
	rank	2	*3*	2	2	1	

that it is not only the constraint type that is important, but also the value specified by the user.

In the accommodation domain we would expect the users to be more rigid regarding temporal and spatial constraints, with factors such as price and star rating also being important.

In question 3 we investigated: alternative facilities - swimming pool (option 2), ignoring the facilities constraint (option 3), alternative destination - Birmingham (option 4), alternative site - cinema (option 5) and alternative type - bed and breakfast (option 6). Only options 1 and 3 do not have statistically significant differences, while options 2 and 5, as expected, have a very low score. The results show that the *disabled access* facility is the most important constraint in the question: users would accept a bed and breakfast or a different city, before considering giving it up. However not all facilities are this important: at the other end of the scale we can imagine room facilities such as *ironing board* or *complimentary newspaper* which usually reflect preference rather than necessity.

In question 5 we investigated: alternative city - Birmingham (option 2), ignoring rating (option 3), alternative rating (option 4), and alternative type - bed and breakfast (option 5). The results suggest that it is useful to inform the users about what hotels are available regardless of star rating, and that a bed and breakfast is less desirable, in this case contrary to the previous question. Therefore, option 4 suggests that when the alternative values are well known (e.g. star ratings for hotels), suggesting an alternative is useful. The last two questions show that the usefulness is influenced not only by the type of the constraints present in the questions, but also by the constrained values.

In question 6, also option 5 which gave the user the full list of hotels and the price charged by each was the most useful (statistically significant). Option 2 was statistically less useful because it just offered a list of hotels, without factoring the price. Picking particular alternatives for constraints (options 3 and 4) scored better than the reference response, but did not prove statistically different.

The analysis presented above shows that suggesting follow-up questions is more useful than just informing the users there were no answers. In most cases, users prefer more general questions which give them information about the available options. Suggesting questions in which one of the initial constraints is changed was not very useful as it may not match the user's preferences. The analysis also revealed that for most constraint types, the usefulness of alternative values depends on the actual values specified in the question, confirming that usefulness depends on the context. This suggests that in order to factor all possible contexts, a system has to automatically learn from users' choices in order to improve its performance.

As a result of the analysis, the prototype was updated to generate responses which help users acknowledge their options in a shortened version allowing preference data to be collected: *The movie 'Matrix' is not on 'tonight' in 'Wolverhampton'. You can either see other **movies**, other **times** (when it is available), or other **destinations** (where it is available).*

To provide such an answer, the system must pre-emptively check that the alternatives proposed actually yield answers. If the user's input matches one of the three follow-up words (e.g. *The movies!*), the system does not have to run the entire pipeline again, but instead, simply returns the answers which have been determined already. In these cases, the interaction has more turns than the initial (question, response) pair.

When two of the constraints cannot be satisfied, the system can say: *The movie 'Matrix' is not on in 'Wolverhampton' (regardless of time). You can see other destinations, or other movies available tonight in Wolverhampton.* However, the generation templates become increasingly complex and the system needs a ranking of constraints to create natural language formulations which are informative and make sense. As result, a rich user interface is perhaps easier to generate.

5 CONCLUSIONS

This paper presented an RDF-based approach for implementing interaction in the QALL-ME framework. An analysis of domain questions revealed that they can be represented as a composition of constraints. This usually takes the form of a conjunction of predicates, as in the following question, *What action movies with Bruce Willis are on in Wolverhampton?* which can be represented as: `hasType(x,qmo:Movie) && hasGenre(x,qmo:action) && hasActor(x,qmo:BruceWillis) && inDestination(x,qmo:Wolverhampton)`. These Boolean predicates correspond to constraints identified in the question by an Entailment Engine which is used to address the problem of language variability. Their truth value can be tested against a data repository by means of inference rules determined by the domain ontology. The Query Generation Web service combines the premises of these rules when generating the WHERE block of a SPARQL query which is used to retrieve the answers to the question, at the same time preserving the semantics of the question interpretation.

The proposed mechanism for allowing user feedback consists of injecting an RDF representation of the question interpretation into the triples of the SPARQL query. Having direct access to this interpretation means that the presentation module can provide feedback, suggest alternative ways in which the question can be asked, and even answer those variations and pre-emptively include the findings in informative answers. The interaction is also more natural from the users' point of view, increasing user satisfaction.

An evaluation of the feedback revealed that simply suggesting follow-up questions is useful, but that usually users want to know all their options in cases where a precise answer cannot be provided. Context is also important, and in order to make competent suggestions the system needs to learn from the choices made by its users. A study is under way to determine whether generating informative answers based on the satisfiability of constraints is useful for the user. A simple feedback loop will be created to allow more interaction based on a single question, via the addition, modification and removal of constraints. This means that the user does not need to pose long or repetitive questions every time they want more information.

REFERENCES

1. Harabagiu, S., Moldovan, D.: Question answering. In Mitkov, R., ed.: Oxford Handbook of Computational Linguistics. Oxford University Press (2003) 560 – 582
2. Hersh, W.: Evaluating interactive question answering. In Strzalkowski, T., Harabagiu, S., eds.: Advances in Open Domain Question Answering. Springer (2006) 431 – 455
3. Rieser, V., Lemon, O.: Does this list contain what you were searching for? Learning adaptive dialogue strategies for interactive question answering. Natural Language Engineering **15**(1) (January 2009) 55–72
4. Quarteroni, S., Manandhar, S.: Designing an interactive open-domain question answering system. Natural Language Engineering **15** (2009) 73–95
5. Sacaleanu, B., Orasan, C., Spurk, C., Ou, S., Ferrandez, O., Kouylekov, M., Negri, M.: Entailment-based question answering for structured data. In: Coling 2008: Companion volume: Posters and Demonstrations, Manchester, UK (2008) 29 – 32
6. Orăsan, C., Dornescu, I., Ponomareva, N.: QALL-ME needs AIR: a portability study. In: Proceedings of Adaptation of Language Resources and Technology to New Domains (AdaptLRTtoND) Workshop, Borovets, Bulgaria (2009) 50 – 57
7. Negri, M., Magnini, B., Kouylekov, M.O.: Detecting expected answer relations through textual entailment. In: Proceedings of 9th International Conference on Intelligent Text Processing and Computational Linguistics, Heidelberg, Germany, Springer (2008) 532–543
8. Ou, S., Pekar, V., Orăsan, C., Spurk, C., Negri, M.: Development and Alignment of a Domain-Specific Ontology for Question Answering. In European Language Resources Association (ELRA), ed.: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco (2008)
9. Magnini, B., Speranza, M., Kumar, V.: Towards interactive question answering: An ontology-based approach. In: Proceedings of the Workshop on Semantic Computing and Multimedia Systems (SCMS 2009), Berkeley, California (September 2009)

IUSTIN DORNESCU
RESEARCH GROUP IN COMPUTATIONAL LINGUISTICS
UNIVERSITY OF WOLVERHAMPTON, UK
E-MAIL: <I.DORNESCU2@WLV.AC.UK>

CONSTANTIN ORĂSAN
RESEARCH GROUP IN COMPUTATIONAL LINGUISTICS
UNIVERSITY OF WOLVERHAMPTON, UK
E-MAIL: <C.ORASAN@WLV.AC.UK>