The QALL-ME Framework: A Specifiable-Domain Multilingual Question Answering Architecture[1]

Óscar Ferrández [a,*], Christian Spurk [b], Milen Kouylekov [c], Iustin Dornescu [d], Sergio Ferrández [a], Matteo Negri [c], Rubén Izquierdo [a], David Tomás [a], Constantin Orasan [d], Guenter Neumann [b], Bernardo Magnini [c], and Jose Luis Vicedo [a]

[a] *University of Alicante, E-03080 Alicante, Spain*
[b] *Deutsches Forschungszentrum für Künstliche Intelligenz - DFKI, Saarbrücken, Germany*
[c] *Fondazione Bruno Kessler, Via Sommarive, 18 - Povo, Trento, Italy*
[d] *Research Group in Computational Linguistics, University of Wolverhampton, UK*

Abstract

This paper presents the QALL-ME Framework, a reusable architecture for building multi- and cross-lingual Question Answering (QA) systems working on structured data modelled by an ontology. It is released as free open source software with a set of demo components and extensive documentation, which makes it easy to use and adapt. The main characteristics of the QALL-ME Framework are: (i) its domain portability, achieved by an ontology modelling the target domain; (ii) the context awareness regarding space and time of the question; (iii) the use of textual entailment engines as the core of the question interpretation; and (iv) an architecture based on Service Oriented Architecture (SOA), which is realized using interchangeable web services for the framework components. Furthermore, we present a running example to clarify how the framework processes questions as well as a case study that shows a QA application built as an instantiation of the QALL-ME Framework for cinema/movie events in the tourism domain.

Key words: Question Answering, Textual Entailment, Natural Language Interfaces, Multilingual Environments

1. Introduction

The exponential growth of digital information requires processes capable of searching, filtering, retrieving and classifying such information. In this context, Question Answering (QA) aims to provide relevant information to end-users as correct answers to an arbitrary question through a search in both unstructured and structured collections of data.

This paper presents the QALL-ME Framework, a reusable architecture skeleton for building multilingual QA systems that answer questions with the help of structured answer data sources from freely specifiable domains. The framework is a free

* Corresponding author. Tel: +34 96 590 3400
Email addresses: `ofe@dlsi.ua.es` (Óscar Ferrández), `Christian.Spurk@dfki.de` (Christian Spurk), `kouylekov@fbk.eu` (Milen Kouylekov), `I.Dornescu@wlv.ac.uk` (Iustin Dornescu), `sferrandez@dlsi.ua.es` (Sergio Ferrández), `negri@fbk.eu` (Matteo Negri), `ruben@dlsi.ua.es` (Rubén Izquierdo), `dtomas@dlsi.ua.es` (David Tomás), `C.Orasan@wlv.ac.uk` (Constantin Orasan), `neumann@dfki.de` (Guenter Neumann), `magnini@fbk.eu` (Bernardo Magnini), `vicedo@dlsi.ua.es` (Jose Luis Vicedo).
[1] This work has been subsidized by the QALL-ME consortium, 6th Framework Research Programme of the European Union (EU), FP6-IST-033860. For more information visit the consortium home page at `http://qallme.fbk.eu/`.

open source software released under the terms of the Apache License 2.0[2]. It is available at `http://qallme.sourceforge.net/` and comes with a set of demo components, which, together with extensive documentation, illustrate the potential of the approach in helping new developers to get started. The framework is highly versatile, and its main characteristics makes it suitable for building multilingual QA architectures when a domain ontology and structured data sources are available. While its components can be easily expanded and replaced with other implementations, domain portability can be achieved by means of an ontology specifying the target domain [23]. Moreover, in order to enhance interoperability and ease of use, the framework seeks to be as far as possible compliant with current standards. The released demo utilizes XML[3] to encode the data structures that store the answers, such structures are accessible via RDF[4] schemas specified by an ontology, in our case developed using OWL-DL[5], and finally to retrieve the answer(s), SPARQL[6] is used as the database query language.

The QALL-ME Framework has been developed within the EU project QALL-ME[7], which pursued the general objective of establishing a shared infrastructure for multilingual and multimodal QA. By employing this framework, the project built a system focused on the tourism domain that has become a concrete business opportunity.

The remainder of the paper is structured as follows: Section 2 and 3 present the framework's main characteristics and the most advanced feature , the use of textual entailment in multilingual QA. Section 4 describes the architecture in detail; a case study is shown in Section 5, followed by related work and conclusions in Sections 6 and 7, respectively.

2. Framework Characteristics

The QALL-ME Framework provides a reusable architecture for multilingual QA systems over structured data belonging to a specific domain. The target domain is modelled by an ontology, used as the main resource to enable multilinguality. The framework takes up-to-date information into consideration by using temporal and spatial reasoning at the time of processing the inquiry. Finally, it finds out the mapping between questions and answers using a novel approach based on textual entailment.

Multilinguality. The QALL-ME Framework supports questions and answers in several different languages. Answers can be retrieved crosslingually, that is, the question and the answers can be in different languages. This capability is possible due to the language identification module, which detects the language of the question and by the spatial context of the inquiry, which is used to infer the language of the answers. Currently, the framework comes with demo components for German, Spanish and English questions, however, its design permits an easy extension by adding the necessary language-dependent components for a new language. For example, in the QALL-ME project we have also implemented an Italian subsystem.

Data Sources. The data sources used to retrieve the answers are usually databases or simply XML documents with a specific structure. With regard to our framework, the data structures have to be accessible via predefined RDF interfaces. These interfaces or RDF schemas are specified by the domain ontology that is used. This implies that the answer data sources are always bound to a certain domain which, however, can be freely specified by exchanging or extending the domain ontology.

Ontology. In order to use the QALL-ME Framework, an ontology containing descriptions both of concepts for the target domain and of possible relations between these concepts has to be provided. The ontology is then used in two ways: (i) as a schema for representing the structure of the answer, in terms of instances of the ontology concepts; and (ii) to cross the language barrier in multilingual QA. By describing the answer by means of the ontology vocabulary, we have a representation that is independent from the original language of the data. Therefore, by creating a mapping from the original question to a query using the ontology vocabulary, we can then apply that query to the answer data and surpass the language barrier.

Regarding the formal language used to encode the ontology, we chose RDF Schema (RDFS) which is recognized as a basic ontology language [2]. However, RDFS is too weak to describe resources in sufficient detail. After it, OIL, DAML, and DAML +

[2] `http://www.apache.org/licenses/LICENSE-2.0.html`
[3] eXtensible Markup Language,`http://www.w3.org/XML/`
[4] The Resource Description Framework (RDF, `http://www.w3.org/RDF`) is a set of specifications designed as a metadata model for describing information.
[5] `http://www.w3.org/TR/owl-guide/`
[6] SPARQL is a standardized query language for RDF data, cf. `http://www.w3.org/TR/rdf-sparql-query`.
[7] Question Answering Learning technologies in a multiLingual and Multimodal Environment, `http://qallme.fbk.eu/`

OIL were designed for more expressive uses [2]. Currently OWL (Ontology Web Language) is the most recent development in standardized ontology languages, endorsed by the World Wide Web Consortium (W3C) to promote the Semantic Web vision. OWL provides three increasingly expressive sublanguages – OWL-Lite, OWL- DL and OWL-Full – for different purposes [16]. In the framework's development, the case study presented next (see section 5) and the project behind this paper (i.e. the QALL-ME project), we have used OWL-DL language because it is much more expressive than OWL-Lite. OWL-DL is based on Description Logics (DL) which means that all conclusions can be guaranteed to be computed and all computations will finish in a finite time so that the classification hierarchy can be computed and the consistencies can be checked. [8]

Spatial-Temporal Context Awareness. The QALL-ME framework is focused on "Spatial-Temporal Context Aware" QA, where the spatial-temporal context in which the question is uttered provides crucial information for the retrieval of the correct answer. Accordingly, all questions are anchored to a certain place in space and time, meaning that every question always has a spatial-temporal context. For instance, using deictic expressions such as "here" or "tomorrow" in a question posed at eight o'clock in Berlin may potentially mean something completely different than the same question posed at five o'clock in Amsterdam. Deictic expressions are solved by algorithms which recognize temporal and spatial expressions in the question and *anchor* relative expressions (e.g. "during the weekend", "the nearest") to absolute expressions (e.g. "May, 22nd", "Unter den Linden, Berlin").

In addition, users may either explicitly indicate the spatial-temporal context in the question (e.g. "Which movies are on tomorrow in Trento?") or leave it implicit, in which case it will be supplied by the system by means of default information (e.g. "Which movies are on?" would be interpreted using "today" and the name of the town where the question is uttered). Therefore, the QALL-ME framework tries to represent the spatial-temporal context information simulating how a human being would interpret such information. Specifically, when it is expressed by deictic expressions, missing or explicitly posed in the query.

Textual Entailment. Behind the scenes, the QALL-ME framework addresses the mapping between a natural language question and a query to the database using Recognizing Textual Entailment (RTE) techniques. RTE techniques allow us to deal with the language variability expressed within the questions through semantic inferences at the textual level. RTE is defined as a generic framework for modelling semantic implications where a certain meaning is described in different ways [10]. RTE components can recognize whether some text T entails a hypothesis H, i.e., whether the meaning of H can be fully derived from the meaning of T. In the context of our framework, H is the minimal form of a question about some topic, and T is the question which has to be answered, cf. section 3.

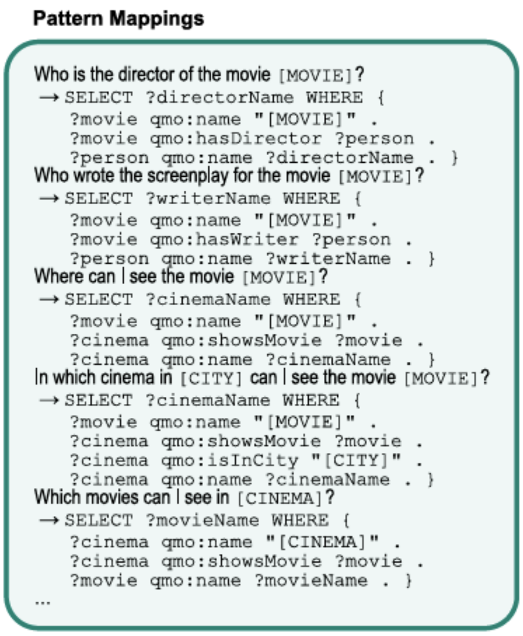
Service Oriented Architecture (SOA). To realize such a demanding QA framework, a flexible dynamic information flow is needed. Consequently, a strong component-oriented perspective for the different QA subtasks has been followed. Based on our experience in large scale system development, the architectural framework is specified from an abstract point of view using generic QA classes that define major input/output representations, and which also covers the specification of the major flow of interaction between the QA components in a declarative way.

We are using SOA as our basic means for the implementation of the QALL-ME core architecture and the concrete QA applications. The QALL-ME SOA describes a set of patterns and guidelines for developing loosely-coupled, highly-reusable QA services that, because of separation of concern between description, implementation and binding, provide both increased interoperability and flexibility in responsiveness to changes and extensions. The functionality of each QA component (see section 4) is realized as a web service and defined via WSDL (Web Service Description Language). The orchestration of the components (i.e., the major information flow) is declaratively defined using BPEL (Business Process Execution Language) and basically maintained by a QA component called *QA-Planner* (see section 4). The orchestrations of different web service implementations create different QA systems and existing web service implementations can easily be reused in different systems. Currently, we are using BPEL to specify standard QA pipelines, however, it is also possible to define more sophisticated information flows, e.g., dynamic selection and orchestration of additional (even competing) components, cf. [22] for more details.

[8] Further details can be found in [26].

3. RTE–based Mapping

The mapping of Natural Language (NL) questions to DataBase (DB) queries is basically controlled via RTE. The core idea is to pre-define a finite bijective mapping between question patterns and corresponding DB query patterns. A question pattern is an NL question string containing variables that correspond to concepts of the ontology, e.g., "Where can I see the movie [MOVIE]?". In an analogous way, a DB pattern is a DB query with corresponding concept variables for the relevant DB slots, e.g., "SELECT ?cinemaName WHERE ?movie qmo:name "[MOVIE]" . ?cinema qmo:showsMovie . ?cinema qmo:name ?cinemaName . ".[9] Note that this mapping actually defines both, the set of answerable DB queries and a frozen semantic interpretation of question patterns. Also note that the ontology is the main interface for semantically aligning these expressions. Fig. 1 shows some further pairs of the mapping defined for the QALL-ME cinema domain.



```
Pattern Mappings

Who is the director of the movie [MOVIE]?
 → SELECT ?directorName WHERE {
    ?movie qmo:name "[MOVIE]" .
    ?movie qmo:hasDirector ?person .
    ?person qmo:name ?directorName . }
Who wrote the screenplay for the movie [MOVIE]?
 → SELECT ?writerName WHERE {
    ?movie qmo:name "[MOVIE]" .
    ?movie qmo:hasWriter ?person .
    ?person qmo:name ?writerName . }
Where can I see the movie [MOVIE]?
 → SELECT ?cinemaName WHERE {
    ?movie qmo:name "[MOVIE]" .
    ?cinema qmo:showsMovie ?movie .
    ?cinema qmo:name ?cinemaName . }
In which cinema in [CITY] can I see the movie [MOVIE]?
 → SELECT ?cinemaName WHERE {
    ?movie qmo:name "[MOVIE]" .
    ?cinema qmo:showsMovie ?movie .
    ?cinema qmo:isInCity "[CITY]" .
    ?cinema qmo:name ?cinemaName . }
Which movies can I see in [CINEMA]?
 → SELECT ?movieName WHERE {
    ?cinema qmo:name "[CINEMA]" .
    ?cinema qmo:showsMovie ?movie .
    ?movie qmo:name ?movieName . }
...
```

Fig. 1. A subset of the NL–DB pairs from our cinema domain, an average of 42 question patterns per language were defined.

As the system response time is directly related to the size of the question pattern set, it would be desirable to reduce this set as much as possible to contain only the essential patterns. This set is referred to as the "Minimal Question Patterns (MQP)" set. Given a set $QP = Q_{p1}, \ldots, Q_{pn}$ of question patterns for a DB query pattern Q, a pattern Q_{pk} belonging to QP is minimal if none of the other question patterns in QP can be derived from Q_{pk} (i.e. is logically entailed by Q_{pk}). Following this statement, a minimal question pattern is defined as a question pattern pertaining to the MQP set.

A new NL question is then processed by the following basic steps:

(i) Question analysis: transforms the input question Q to an NL pattern P_Q;
(ii) Identification of NL pattern: uses RTE to determine the corresponding NL–DB pair pattern (P_{NL}, P_{DB}) by testing whether P_{NL} is entailed in P_Q;
(iii) Instantiation of DB pattern: instantiates P_{DB} using corresponding extracted entities from Q.

For example, the question analysis component computes for the NL question "Which cinemas show the movie Dreamgirls tonight?" the NL pattern "Which cinemas show the movie [MOVIE] tonight?" and the entity "[MOVIE]=Dreamgirls". Through textual entailment the pair ("Where can I see the movie [MOVIE]?", "SELECT ?cinemaName WHERE ?movie qmo:name "[MOVIE]" . ?cinema qmo:showsMovie . ?cinema qmo:name ?cinemaName . ") is selected from the mapping and the corresponding DB-pattern is instantiated as "Where can I see the movie Dreamgirls?", "SELECT ?cinemaName WHERE ?movie qmo:name "Dreamgirls" . ?cinema qmo:showsMovie . ?cinema qmo:name ?cinemaName . ", using the extracted entity from the input question. To complete the retrieval procedure, we simply execute the query to obtain the correct answer(s).

Note that the main interface for the RTE engine is the NL pattern syntax. Therefore, how exactly, the RTE engine is realized is not important in principle. Actually, in the QALL-ME project we have experimented with different approaches, ranging from simple Bag-Of-Words approaches to a sophisticated Machine Learning algorithm exploiting deep grammatical parsing of the NL patterns, cf. [30].

4. Framework Architecture

The QALL-ME Framework workflow is managed by the *QA Planner*, which is in charge of orchestrating the web service components and thus passing the

[9] As the QALL-ME Framework assumes the answer data to be represented in RDF, it is only natural to use SPARQL as the database query language.

input question through the whole QA system until an answer is found. Fig. 2 depicts a conceptual view on this workflow in the planner.

As shown in the figure, the *QA Planner* processes the user's question until an answer is achieved. To do this, the *QA Planner* receives five input parameters: the asked question, two parameters belonging to the spatial-temporal context (current time and user's location), a collection of facts stored into a database of answers, and a collection of pattern mappings stored into a pattern repository containing the minimal question patterns and their corresponding database queries. This repository is used by the RTE components to establish the corresponding inferences with the input question and can be either created manually or by a suitable learning subsystem.

The single output parameter of the *QA planner* is the found answer.

4.1. *System Components*

The workflow represents both the monolingual and the crosslingual system's run; the *QA Planner* is the responsible service for selecting appropriate components on the way to the answer.

Regarding the components, they are of three different kinds, shown and annotated with relevant comments in Fig. 2:

– *Language-specific*: components that are implemented for each language that the QA system will be able to handle for input questions.
– *Location-specific*: different implementations for each location at which the QA system will be able to answer questions. This is in line with the spatial anchoring of questions.
– *System-wide*: unique implementations for the whole QA system, i.e., these implementations have language- and location-independent functionality and the *QA Planner* uses them for all input questions.

The first step implemented by the *QA Planner* is to create a *Question Object* (called *QObj*) which will be used throughout the QA process. At the beginning of the process the *QObj* contains the input question together with the contextual information related to the question. The following components will then be used to extend the *QObj*:

(i) *Language Identification*: automatically recognises the source language of the question and adds it to the *QObj*. Based on this information and on the context of the question the *QA Planner* selects the appropriate location and the language-dependent components.
(ii) *Entity Annotation*: annotates entity names in the question depending on available entity names contained in the database of answers. The entity types used correspond to those of the domain of the QA system and they have to be specified as concepts in the ontology.
(iii) *Term Annotation*: annotates language-specific terms that are relevant for the domain. It differs from the *Entity Annotation* component in that the annotation of named entities is usually language-independent, whilst relevant terms are used to express concepts only in certain languages. Consider, for example, hotel facilities: terms for concepts like TV, swimming pool, safe, etc. are represented differently in different languages, e.g., the English "hair dryer" is equivalent to the German "Fön".
(iv) *Temporal Expression Annotation*: to complete the annotation process, temporal expressions have to be annotated, too. Obviously, such expressions are language-specific and will depend on the temporal context. The previous three components – apart from annotating – also convert each annotated entity, term or expression into a canonical (i.e. normalized), language-independent form. For instance, a temporal expression is normalized into a common format like TIMEX2[10] or ISO 8601. By using these canonical forms, it is much easier for the other components to translate such expression into a suitable search query.
(v) *Query Generation*: provides a proper query to the database, in order to find an answer to the input question. This module chooses a suitable language-dependent RTE component, which returns the semantically inferred minimal question pattern and its corresponding query to the database, to be applied to the available data. Depending on the its implementation, the RTE component can also lead to the selection of more than one minimal question pattern, each one matching different parts of the input question. In this case, the SPARQL queries can then be composed into a single query for answer retrieval [20].
(vi) Finally, at the end of the process, two distinct situations may occur: (1) if no query is gener-

[10]`http://timex2.mitre.org`

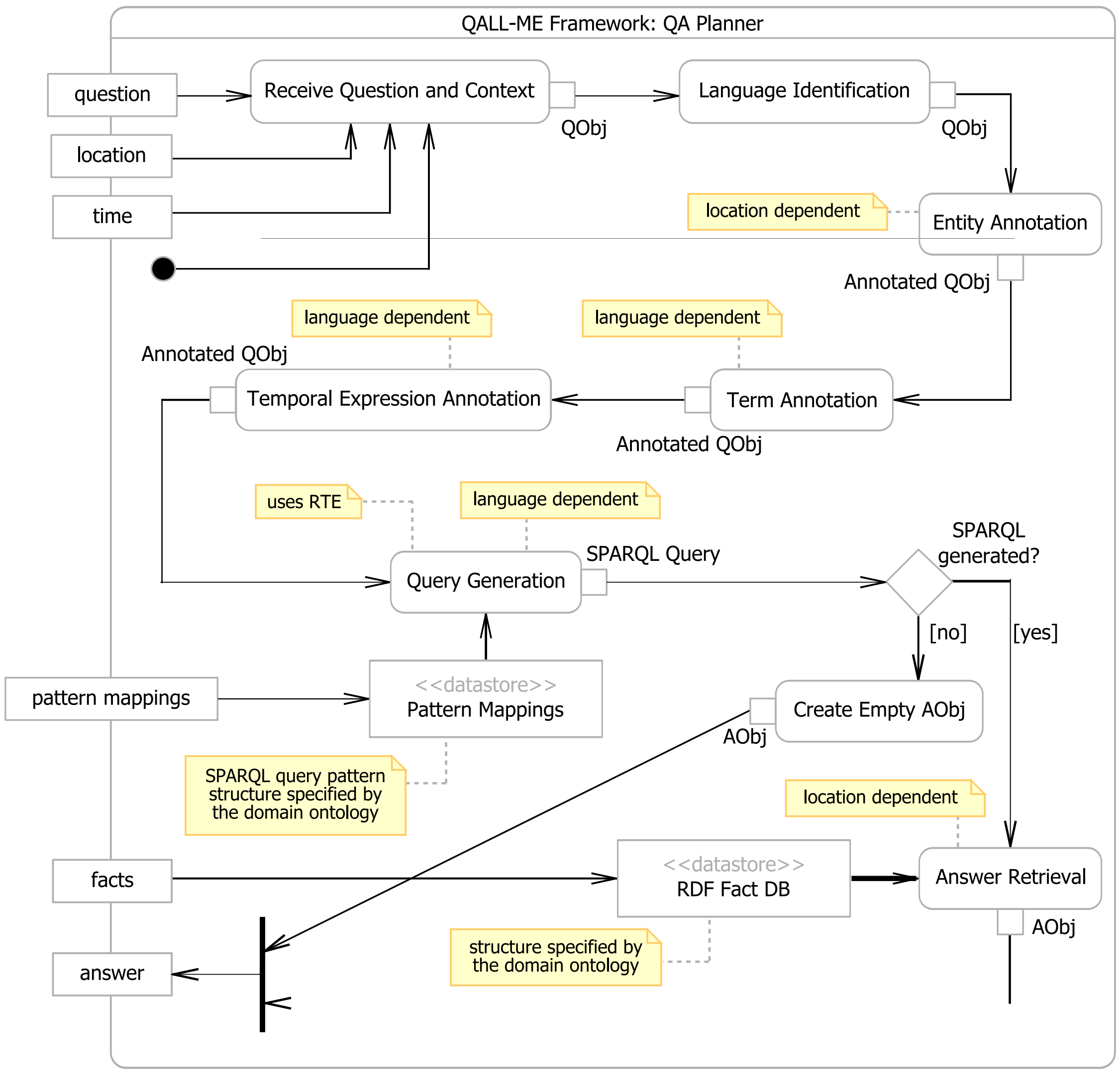


Fig. 2. The QALL-ME Framework: conceptual view of the QA workflow in the QA planner as a UML 2 activity diagram.

ated, then the user has probably posed an out-of-domain question and the system asks the user to re-formulate the question; or (2) the generated query is fed into the location specific *Answer Retrieval*, component which corresponds to the spatial context of the question. This component will then retrieve the answer(s) to be output by the system.

4.2. *A Walk-Through Example*

To clarify the procedure carried out by the *QA Planner*, we present a running example showing how a new question is processed through the appropriate framework components until an answer is reached for the corresponding input.

Input Question:
"Where is the movie Matrix being shown today?"
Context Information
Loc="Alicante" Time="2010-08-06T11:34:56"

(1) The *QA Planner* receives the input information (i.e. question and context) and selects the corresponding location-dependent components. In this case the *Entity Annotation* and the database of facts/answers will be taken from the Spanish components.
(2) The *Language Identification* component detects that the input question is in English, therefore the

language-dependent components such as *Term Annotation*, *Temporal Expression Annotation*, *Query Generation* and consequently the RTE engine and the repository of minimal question patterns will be taken from English components.
(3) The Spanish *Entity Annotation* component takes the input question and creates an annotated *QObj* regarding the entities found: "Where is the movie [MOVIE] being shown today?".
(4) The English *Term Annotation* component receives the *QObj* and recognises terms expressing relevant aspects for the answer retrieval (such as cinema facilities). In this example no terms are found for the input question.
(5) The English *Temporal Expression Annotation* component detects and normalizes temporal items: "Where is the movie [MOVIE] being shown [2010-08-06]?".
(6) The English *Query Generation* component, using the corresponding RTE engine, achieves a mapping to one of the minimal question patterns from the repository, collecting also the SPARQL query for answer retrieval. Minimal Question Pattern found ⇒ "Where can I see [MOVIE]?".
(7) The Spanish Answer Retrieval component launches the SPARQL query associated with the minimal question pattern and retrieves the answer(s) from the Spanish database of facts. Note that this components also takes into account information about terms and temporal expressions stored in the *QObj*. Answer(s) found ⇒ CINEMA: Colci.

5. Case Study

Although the framework components can be instantiated and applied to any domain modelled by an ontology [23], within the QALL-ME project a case study has been built and tested for the tourism domain. This case study shows an end-to-end multilingual QA system working on cinema/movie data for English, German, Italian and Spanish. The next paragraphs describe the instantiation and evaluation of the QALL-ME Framework for the above mentioned domain.

The first step was to develop the domain ontology, modelling the tourism domain. This ontology was presented in [24], and it is freely available.[11] The ontology covers important aspects of the tourism industry, including tourist destinations, sites, events and transportation, and it provides a common vocabulary for the domain as well as a computerized specification of the meaning of terms used in the vocabulary. OWL-DL was selected as the encoding language to implement the ontology.

The final version of the ontology contains 261 classes (concepts), 55 datatype properties which indicate the attributes of the classes, and 55 object properties which indicate the relationships among the classes. Fig. 3 shows a schematic view of a portion of the ontology including its classes and their most important relationships.

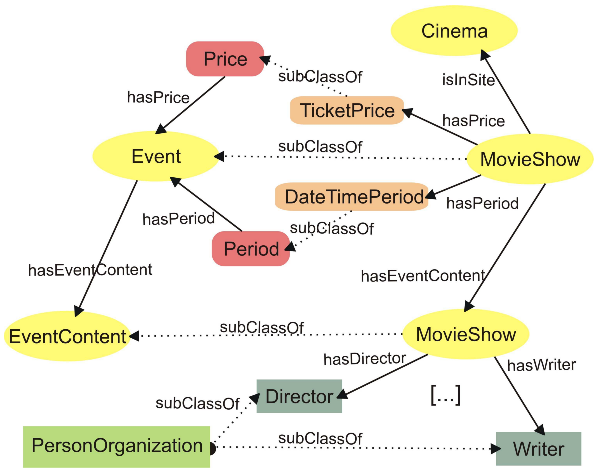


Fig. 3. A portion of the QALL-ME tourism ontology.

The classes were categorized into 15 top-level classes falling into three categories:[12]

(1) **Main classes** refer to the most important concepts in the tourism domain, including `Country`, `Destination`, `Site`, `Transportation`, `EventContent` and `Event`. The instances of the main classes are concrete countries, destinations, sites, transportation vehicles, event contents (e.g. *movie*) and occurrences of event contents (e.g. *movie show*).
(2) **Element classes** refer to the elements of the main classes or the elements of other element classes, including `Facility`, `Room`, `Person & Organization`, `Language` and `Currency`. For example, an instance of the class `GuestRoom` (subclass of the `Room` class) is an element of an `Accommodation` instance, and an instance of `GuestRoomFacility` is an element of a `GuestRoom` instance.

[11] `http://qallme.fbk.eu/index.php?location=ontology`

[12] Further details about the ontology classes, attributes and relationships can be found in [24] and [26].

(3) **Attribute classes** refer to the packages of a group of attributes for the main classes or element classes, including `Contact`, `Location`, `Period` and `Price`. For example, a `Contact` instance contains all kinds of contact information (e.g. *contact person*, *telephone*, *fax*, *email*, and *website*) for a `Site` instance.

Once the ontology was created, the next step was to generate the set of minimal questions patterns needed for the RTE component (see section 3). With the aid of the ontology conceptualizations, we generated possible scenarios for making inquiries (e.g., "you are in [DESTINATION] and you want to see the movie [MOVIE]"). These scenarios were presented to potential users and they were asked to freely formulate natural language requests according to such scenarios. The user requests were collected in the QALL-ME benchmark [5], which includes questions for each language involved in the QALL-ME project, i.e., German, English, Italian and Spanish.

We generated automatically at least one scenario for each ontology relation and concept attribute susceptible to be asked. Such scenarios were linked with their corresponding queries to the database, and the requests made by the users were used to build the minimal question pattern set. [13]

As for the evaluation of the case study, we considered a subset of the QALL-ME tourism ontology, including concepts, relations and minimal question patterns specific of the Cinema subdomain. A total number of 54 scenarios were generated for this subdomain, and on average 400 user queries were collected for each language. Consequently, we populated the ontology only with cinema and movie information by local data providers. The information provided was converted into RDF answer databases, covering movie events for a specific period of time as well as for specific locations.

Regarding the knowledge base of minimal question patterns, we extracted from the aforementioned scenarios an average number of 42 patterns per language. Such patterns were chosen with the aim of covering most of the relations and concepts modelled by the ontology, some examples of these patterns are "What is the address of the cinema [CINEMA]?", "At what time is [MOVIE] shown in the cinema [CINEMA]?", and "In which cinemas can I see [MOVIE]?", to name but a few.

[13] Note, however, that depending on specific purposes, the set of minimal question patterns can be created in any other way, too. Using ontology scenarios, as we did, generating them by a manual creation procedure, or implementing/designing new methods for learning/acquiring the minimal question patterns from the ontology and/or other sources.

In our prototype-demo [27], an evaluation focused on the QALL-ME benchmark was carried out. In addition, we decided to evaluate the system on-field, by using a set of questions posed by real users in a real situation. This has the advantage of using questions which are different from the ones that were used to the develop the system and which were collected in a more realistic setting. In the on-field study we gathered independently-generated user questions from the cinema domain. The users received only a brief description of the system – just to make them aware of its capabilities. The final set of questions contained 304 cinema questions from the Trentino region (Italy); each of these questions was then translated, so that a parallel corpus for German, English, Italian and Spanish could be built. Since the questions were first formulated by Italian speakers, the entities appearing in them were Italian, and a common *Entity Annotator* had to be used by all language-dependent subsystems.

Table 1 shows the accuracy achieved by each language subsystem, i.e. the percentage number of answers considered correct, since they provided to the user the information solicited. Furthermore, since the RTE components are the core components of our framework, we also show their performance in returning correct associations between the input questions and minimal question patterns (and consequently the retrieval of the answer(s)). To obtain the RTE component performances, we did not consider the evaluation questions that were wrongly answered due to errors in other components or errors not related directly to wrong RTE associations (see last column in Table 1).

Language	**Accuracy**	**RTE component performance**
English	60.85%	81.3%
German	67.76%	84.4%
Spanish	77.96%	92.2%
Italian	85.00%	90.0%
all (∅)	72.89%	86.97%

Table 1
Evaluation results for each language-dependent subsystem over the on-field question dataset.

The differences of the accuracy results for each language are mainly due to the different implementations and performance of the language-dependent components (such as the RTE engines and the temporal annotators) and the domain coverage of the minimal question pattern set for each language, which is directly related to the variety of user questions collected in the QALL-ME benchmark.

An analysis of the errors revealed several situations: (i) *yes/no questions* such as "Is Gomorra playing in any cinema in Trentino?" cannot be answered by the system in its current state, a total of 19 (6.25%) yes/no questions were found within the question set; (ii) *errors in entity and/or temporal annotations* result in a misclassification of questions and a wrong retrieval of results, on average 20.3 (6.67%) questions per language were wrong answered due to this kind of errors; (iii) *missing patterns* in the previously defined set of minimal questions patterns that would be needed to answer an input question. In numbers, 29.6 (9.73%) questions on average per language could not be answered due to this circumstance; and (iv) *wrong entailment association* where the entailment components failed to find the correct entailed question pattern, which represents 32.3 (10.62%) questions on average per language from the question evaluation set.

In order to calculate the performance of the RTE components (last column of Table 1) we created another question evaluation set by removing the questions that were wrongly answered by *yes/no*, *entity/temporal annotation* and *missing pattern* errors. It resulted in a set of 235.1 questions on average per language, and running the system using this question set we are able to measure the accuracy of the RTE components in mapping the input questions to the correct minimal question pattern, which also retrieves the information solicited. Values between 81% and 92% suggest that the use of the RTE components in our framework is appropriate, although obviously it is also dependent of the quality and representativeness of the minimal pattern question set.

The demo components released with the free open source software package of the QALL-ME Framework are partly based on this case study, providing reduced versions of both our RDF answer databases and the minimal question repositories.

6. Related Work

Although the literature on QA systems is extensive, the availability of these systems as open-source is quite limited. Current open-source distributions can be classified into two broad groups depending on the type of information source used to find answers. Textual-based QA systems retrieve answers from the Web and/or other textual sources. Systems like Qanda[14], OpenEphyra[15], Aranea[16], QANUS[17] and OSQA[18] pertain to this category. In contrast, QA system over structured data, traditionally addressed through Natural Language Interfaces (NLIs), like Aqualog[15], NLBean[19] or SQ-Hal[20], obtain the answers from structured information sources organized under a previously defined data model.

The QALL-ME Framework is a reusable architecture to create multilingual QA systems accessing structured data and consequently, falls into the second category. Table 2 shows the main features of these distributions, including our framework, in terms of the type of information source used and their multilingual and spatial-temporal context capabilities. Only the QALL-ME framework develops a full multilingual setup which allows a direct processing of questions in different languages. This fact allows users to access all the information available in the ontology posing questions in any of the included languages. In contrast, the rest of the distributions are English-based monolingual systems. Although these systems could be adapted to multilingual environments by adding translation capabilities, it has been proven through experimentation (see CLEF QA track series[21]) that these kinds of multilingual approaches seriously reduce the system performance in comparison with its monolingual version. Offering implicit spatial-temporal context capabilities is another distinctive characteristic of the QALL-ME Framework. While some of the referred systems are able to detect and use explicit time and location expressions appearing in a question such as dates or city names (explicit spatial-temporal processing), only the proposed framework permits, in addition, to convert implicit spatial-temporal references such as "this evening" or "near hear" into specific time and place coordinates by taking into account the time and location information anchored to the questions when they are posed.

[14] `http://sourceforge.net/projects/qanda/`
[15] `http://www.ephyra.info/`
[16] `http://www.umiacs.umd.edu/~jimmylin/downloads/Aranea-r1.00.tar.gz`
[17] `http://www.comp.nus.edu.sg/~junping/docs/qanus.pdf`
[18] `http://www.osqa.net/`
[19] `http://www.markwatson.com/opensource/`
[20] `http://www.csse.monash.edu.au/hons/projects/2000/Supun.Ruwanpura/`
[21] `http://www.clef-campaign.org/`

System	Information Source	Languages	Spatial-Temporal Context
Qanda	Text/Web	Monolingual	Explicit
OpenEphyra	Text/Web	Monolingual	Explicit
Aranea	Text/Web	Monolingual	Explicit
QANUS	Text/Web	Monolingual	Explicit
OSQA	Text QA pairs	Monolingual	Explicit
NLBean	Relational DB	Monolingual	Explicit
SQ-Hal	Relational DB	Monolingual	Explicit
AquaLog	Ontology	Monolingual	Explicit
QALL-ME	Ontology	Multilingual	Explicit & Implicit

Table 2
QA open-source distributions.

Focusing on NLIs development, the rest of this section gives an overview of relevant related works on NLIs, emphasizing the differences between the QALL-ME Framework and previous approaches.

In this analysis, we distinguish between three categories of interfaces: *i)* Full Natural Language Interfaces; *ii)* Restricted Natural Language Interfaces; and *iii)* Instance-based methods.

Full NLIs. QA over structured data has been traditionally addressed through a deep analysis of the question in order to reconstruct its logical form, which is then translated into the query language of the target data [1,25,6,17]. This approach implies a complex mapping between linguistic objects (e.g. lexical items, syntactic structures) and data objects (e.g. concepts and relations in a knowledge base). As shown by [12], full NLIs are often preferred by users over other solutions to access structured information (*e.g.* forms-based, and graphical query language interfaces). Several studies, however, agree on their limitations in terms of portability, due to the high costs associated with their development, configuration, and customization to new domains [9,1,7,8]. Despite recent efforts in developing authoring tools to support NLIs configuration [18], the mapping between language and data structures still requires intensive manual work by domain experts and language engineers. This represents a bottleneck in the realization of large scale and portable NLIs.

Compared to traditional full NLIs, the QALL-ME Framework allows for a considerable reduction in the overall porting costs. As shown in Section 2, leveraging the notion of textual entailment the problems of finding logical representations of questions and then translating them into database queries is bypassed through semantic inferences at the textual level. Since it does not require explicit mappings between language expressions and the data objects stored in the target database, the textual entailment-based approach allows the system to effectively cope with the fact that linguistic phenomena are usually independent from the database schema. It is also worth mentioning that, as it is implemented in the QALL-ME Framework, such a solution presents advantages in terms of flexibility and adaptability. As regards flexibility, SOA allows new textual entailment components to be plugged in easily (*e.g.* for a new language, as done for Italian) with no impact on other components. As regards adaptability, the textual entailment-based approach allows experimentation with different solutions to the problem of QA over structured data. As an example, still in the QALL-ME scenario, [20] presents an alternative approach casting the question analysis problem as a Relation Extraction task. Though quite different with respect to the strategies adopted for the other languages, the Italian subsystem based on this approach has been successfully integrated in QALL-ME.

Controlled NLIs. Controlled NLIs have been proposed as a simple solution to avoid the issues raised by language variability on the one side, still hiding the complexity of formal languages on the other side. In such a framework, users' requests are composed using a controlled language with terminological and grammatical restrictions. Among others, the system presented in [4] uses "Attempto Controlled English", an unambiguous subset of English which can be automatically transcribed into a triple-based semantic web language to query ontologies. A controlled natural language interface is also proposed by [21] as an effective way to translate users' requests into SQL/Temporal, a query language for temporal databases. Though suitable to tackle the language variability problem, and effectively fill the gap between users' information needs and valid queries to access the data, controlled NLIs present some limitations. First, the adoption of a controlled language imposes a cost on the user since the language has to be learned. As pointed out by [4], though much cheaper than learning logic and the query language of the target data, the cost of training users on the controlled language is not negligible. Also, controlled languages feature limited portability since manual adaptation of the rewrite rules is often required to use them with a new ontology or a new knowledge base.

The QALL-ME Framework overcomes both the aforementioned limitations of controlled NLIs. On one side, the system is robust enough to allow for a natural and user-friendly interaction, exploiting the full potential of unrestricted natural language queries. On the other side, scalability and portability of the approach come at the limited cost of providing a relatively small set of minimal questions lexicalizing relations in a new target domain ontology, as described in Section 2.

Instance-based methods. Instance-based methods for QA are based on learning answering strategies (S) directly from clusters of similar training question-answer pairs ($Q - A$). Thus, answering a new question (Q_n) consists on performing a question to question matching (Q-to-Q) to find the training question pair ($Q_j - A_j$) whose Q_j part is most similar to Q_n. Next, the answer to Q_n is retrieved by applying the strategy S_j associated to the cluster Q_j pertains to.

Q-to-Q matching performance is crucial for instance based QA systems since it needs to address all the issues regarding language variability in the input question in order to match exactly the question-answer pair that retrieves the correct answer. Q-to-Q matching methods have been applied to both QA system categories: text- and structured-data-based.

In *text-based QA*, the adoption of such approaches has been motivated by the need for principled, statistically based, easily re-trainable, and language independent QA systems that take full advantage of large amounts of training data. To these aims, the system described in [14] considers a multi-dimensional space, determined by features (*e.g.* lexical n-grams, parse trees elements) extracted from training examples. Training questions are represented as data points in this space, allowing the construction of a set of neighborhood clusters, and a corresponding model for each of them. At the test stage, in order to perform the Q-to-Q matching, each model is applied to the input questions to produce its own set of candidate answers. The FAQFinder system described in [29] focuses on retrieving information from a Frequently Asked Questions (FAQ) archive, containing question-answer pairs where questions are pre-answered and compiled by domain experts. The basic Q-to-Q matching method adopted by the system relies on four complementary metrics to judge the similarity between a user question and an FAQ question: *i)* cosine similarity; *ii)* word overlap; *iii)* WordNet-based semantic similarity; and *iv)* question type similarity. Similarly, in [11] and [31], Q-to-Q matching mechanisms are applied over community-collected QA archives containing peoples' answers to other peoples' questions posed on the Web. To solve the problem of word mismatches between users' questions and question-answer pairs in the archive, the matching process relies on a translation-based approach, which implicitly expands queries using translation probabilities [3]. More recently, [19] presented a speech-based QA system using Q-to-Q matching to retrieve answers from two different types of information sources: *i)* a large repository of question-answer pairs collected from the Web for *static* questions; and *ii)* trusted content aggregator websites for *dynamic* questions. Q-to-Q matching is performed by combining several metrics including TF-IDF scores and string comparison metrics such as Levenshtein edit distance and n-gram overlap.

As regards *QA over structured data*, Sneiders [28] proposed a shallow Q-to-Q matching method based on a keyword comparison technique called Prioritized Keyword Matching. Such a method relies on simple pattern matching of the input question to a number of question templates (dynamic parameterized questions about events in a large city). Finally, a similar pattern-based technique is used in the AquaLog [15] system as a core mechanism to translate natural language questions into combinations of ontology-compliant triples used to query an ontology. Regular expressions are used to process input requests, and classify them into 14 question types, or intermediate representations. The correct classification of the question into one of the "prototypical" cases known by the system is crucial for correct answer retrieval.

All these approaches are directly relevant and in various aspects similar to our work, which can be classified as an instance-based method as well. However, also in this case the strategies adopted in the QALL-ME Framework present novelties and advantages over previous works. First, textual entailment-based Q-to-Q matching is an original and more flexible solution compared to the mapping techniques previously described. Compared to word-overlap, cosine similarity, question type similarity, and pattern-based methods, the entailment framework suggests sophisticated inferences in order to deal with language variability, which in theory is a potential improvement. In addition, it offers lots of room to integrate advanced algorithms and knowledge derived from a variety of sources (*e.g.*

ontologies, the Web, large corpora), which have not been fully exploited yet. A second advantage of the matching strategies implemented in the QALL-ME Framework is represented by the possibility of composing the output of partial matches. Previous works try to find a *perfect match* between users' questions and the questions stored in the archive. Hence, for successful answer retrieval, the archive should contain at least one question expressing the same combination of constraints (*i.e.* ontology relations) expressed by the input question. In contrast, textual entailment-based Q-to-Q matching allows the collection of multiple entailed minimal question patterns (each representing an ontology relation), and combines the associated SPARQL queries into a single query to the target database. The two different strategies have different impact on the system's scalability and portability. Extending, or porting traditional instance-based methods to new domains (or languages) requires the collection of large quantities of prototypical questions representing all the possible users' information needs in the new domain (or language).

Porting the QALL-ME Framework only requires the availability of at least one sample minimal question pattern lexicalizing each relevant domain relation. Note, however, that the performance of a QA system based on the QALL-ME Framework will be conditioned by the ontology, since just concepts and semantic relationships modelled by the ontology will be able to be answered.

7. Conclusions and Future Work

The main contribution of this research is the release of an open source software package that implements a reusable, multilingual and context-aware QA Framework, working on structured data from specifiable domains.

While from a broad perspective the QALL-ME framework is focused on solving the QA problem on restricted domains, its main characteristics cover many other aspects, which might go beyond the interest of the QA community and attract a wide NLP audience. For instance, the framework is based on *SOA*, which provides interoperability and flexibility realized by using web services for the implementation of the potentially interchangeable components. The framework *Multilinguality* is achieved by an ontology modelling the domain, which also represent a framework limitation since an available ontology has to be previously specified. At the time of posing questions, the framework also carries out *temporal* and *spatial reasoning*, which leads to up-to-date and coherent answers. And finally, the task of finding the correct mapping between question and answer is carried out with *RTE-based* components. Specifically, the scientific contribution of exploiting textual entailment for question interpretation comes from our idea that featuring the capability of composing queries mapping a question to multiple question patterns goes beyond calculating a similarity score between questions. Therefore, textual entailment provides nicer semantic interpretations to our framework, although further investigations are needed in this field in order to achieve a solution for all the different situations the system faced with.

Furthermore, the distribution of the framework as an open-source allows us to share the results of our research work and others to repeat and investigate future improvements of our experiments.

As for future research directions, automatic acquisition of minimal question patterns is one of the most promising directions emerged at the end of the project. In this direction, we started preliminary experiments aiming at deriving minimal question patterns using corpora and the domain ontology. With regard to other aspects, more efforts have to be undertaken in Interactive QA for the development of modules capable of providing rich and natural answers and dialogue functionalities for enhancing user-system interaction. Furthermore, the framework characteristics allow an easy extension or replacement of its components: for instance, since research on textual entailment is constantly improving, such improvements can be added into our RTE components; or they can be replaced by freely available textual entailment recognizers such as the EDITS system[13], which is another outcome of the QALL-ME project.

References

[1] I. Androutsopoulos, G. D. Ritchie, P. Thanisch, Natural Language Interfaces to Databases - An Introduction, CoRR cmp-lg/9503016.

[2] S. Arroyo, R. Lara, Y. Ding, M. Stollberg, D. Fensel, Semantic Web Languages. Strengths and Weakness, in: International Conference in Applied Computing, 2004.

[3] D. Bernhard, I. Gurevych, Combining Lexical Semantic Resources with Question & Answer Archives for Translation-Based Answer Finding, in: Proceedings of

the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Suntec, Singapore, 2009.

[4] A. Bernstein, E. Kaufmann, A. Göhring, C. Kiefer, Querying Ontologies: A Controlled English Interface for End–Users, in: Proceedings of the 4th International Semantic Web Conference, ISWC'05, LNCS 3729, Galway, Ireland, 2005.

[5] E. Cabrio, M. Kouylekov, B. Magnini, M. Negri, L. Hasler, C. Orasan, D. Tomas, J. L. Vicedo, G. Neumann, C. Weber, The QALL-ME Benchmark: a Multilingual Resource of Annotated Spoken Requests for Question Answering, in: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco, 2008.

[6] H. C. Chan, J. Lim, A Review of Experiments on Natural Language Interfaces, in: Advanced Topics in Database Research, Vol. 2, 2003, pp. 55–71.

[7] P. Cimiano, P. Haase, J. Heizmann, Porting natural language interfaces between domains: an experimental user study with the ORAKEL system, in: Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI 2007), Honolulu, Hawaii, USA, 2007.

[8] P. Cimiano, M. Minock, Natural Language Interfaces: What is the Problem? A data-driven quantitative analysis, in: Proceedings of the International Conference on Applications of Natural Language to Information Systems (NLDB), Saarbruecken, Germany, 2009.

[9] A. Copestake, K. S. Jones, Natural Language Interfaces to Databases, Knowledge Engineering Review 5 (4) (1990) 225–249.

[10] I. Dagan, O. Glickman, Probabilistic textual entailment: Generic appied modelling of language variability, in: Proceedings of the PASCAL Workshop on Learning Methods for Text Understanding and Mining, Grenoble, France, 2004.

[11] J. Jeon, W. B. Croft, J. H. Lee, Finding Similar Questions in Large Question and Answer Archives, in: Proceedings of the CIKM 2005, 14th ACM international Conference on Information and Knowledge Management, Bremen, Germany, 2005.

[12] E. Kaufmann, A. Bernstein, How Useful are Natural Language Interfaces to the Semantic Web for Casual End-users?, in: Proceedings of ESWC 2007, LNCS 4825, Busan, Korea, 2007.

[13] M. Kouylekov, M. Negri, An Open-Source Package for Recognizing Textual Entailment, in: Proceedings of the ACL-2010 System Demonstrations, Sweden, 2010.

[14] L. V. Lita, J. G. Carbonell, Instance-Based Question Answering: a Data-driven Approach, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 2004.

[15] V. Lopez, V. S. Uren, E. Motta, M. Pasin, AquaLog: An ontology-driven question answering system for organizational semantic intranets, Journal of Web Semantics 5 (2) (2007) 72–105.

[16] D. L. McGuinness, F. van Harmelen, OWL Web Ontology Language Overview (2003).
URL `http://www.w3.org/TR/owl-features/`

[17] M. Minock, A Phrasal Approach to Natural Language Interfaces over Databases, in: Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems, LNCS 3513, Alicante, Spain, 2005.

[18] M. Minock, P. Olofsson, A. Näslund, Towards Building Robust Natural Language Interfaces to Databases, in: Proceedings of the 13th International Conference on Applications of Natural Language to Information Systems, London, UK, 2008.

[19] T. Mishra, S. Bangalore, Speech-driven Access to the Deep Web on Mobile Devices, in: Proceedings of the ACL 2010 System Demonstrations, Sweden, 2010.

[20] M. Negri, M. Kouylekov, Question Answering over Structured Data: an Entailment-Based Approach to Question Analysis, in: Proceedings of the International Conference RANLP-2009, Borovets, Bulgaria, 2009.

[21] R. Nelken, N. Francez, Querying Temporal Databases Using Controlled Natural Language, in: COLING 2000, 18th International Conference on Computational Linguistics, Saarbrücken, Germany, 2000.

[22] G. Neumann, C. Spurk, B. Sacaleanu, The QALL-ME Architecture – Design Issues and QA Framework, public project deliverable 2.1.: Specification of QA framework and service policies (http://qallme.fbk.eu/) (2007).

[23] C. Orasan, I. Dornescu, N. Ponomareva, QALL-ME Needs AIR: a Portability Study, in: Adaptation of Language Resources and Technology to New Domains (AdaptLRTtoND) Workshop, 2009.

[24] S. Ou, V. Pekar, C. Orasan, C. Spurk, M. Negri, Development and Alignment of a Domain-Specific Ontology for Question Answering, in: Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco, 2008.

[25] A.-M. Popescu, O. Etzioni, H. A. Kautz, Towards a theory of natural language interfaces to databases, in: Proceedings of the 2003 International Conference on Intelligent User Interfaces, Miami, FL, USA, 2003.

[26] QALL-ME Consortium (http://qallme.fbk.eu/), QALL-ME: Semantic Web and Data Access, Tech. rep., FBK, University of Wolverhampton, University of Alicante, DFKI, Comdata, Ubiest, Waycom (2009).

[27] B. Sacaleanu, C. Orasan, C. Spurk, S. Ou, Ó. Ferrández, M. Kouylekov, M. Negri, Entailment-based Question Answering for Structured Data, in: Coling 2008: Companion volume: Demonstrations, Coling 2008 Organizing Committee, Manchester, UK, 2008.

[28] E. Sneiders, Automated Question Answering Using Question Templates that Cover the Conceptual Model of the Database, in: Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems, Stockholm, Sweden, 2002.

[29] N. Tomuro, S. L. Lytinen, Retrieval Models and Q and A Learning with FAQ Files, in: M. T. Maybury (ed.), New Directions in Question Answering, AAAI Press, 2004, pp. 183–202.

[30] R. Wang, G. Neumann, Recognizing Textual Entailment Using a Subsequence Kernel Method, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, Vancouver, Canada, 2007.

[31] X. Xue, J. Jeon, W. B. Croft, Retrieval Models for Question and Answer Archives, in: Proceedings of the the 31th International ACM SIGIR Conference (SIGIR'08), Singapore, 2008.