QAKiS @ QALD-2

Elena Cabrio, Julien Cojan, Bernardo Magnini, Fabien Gandon, Alberto Lavelli

QAKiS @ QALD-2

Elena Cabrio[1], Alessio Palmero Aprosio[2,3], Julien Cojan[1],
Bernardo Magnini[2], Fabien Gandon[1], and Alberto Lavelli[2]

[1] INRIA, 2004 Route des Lucioles BP93, 06902 Sophia Antipolis cedex, France.
{elena.cabrio, julien.cojan, fabien.gandon}@inria.fr
[2] FBK, Via Sommarive 18, 38100 Povo-Trento, Italy.
{lavelli, magnini}@fbk.eu
[3] Università degli Studi di Milano, Via Festa del Perdono 7, 20122 Milano, Italy.
alessio.palmero@unimi.it

Abstract. We present QAKiS, a system for Question Answering over linked data (in particular, DBpedia). The problem of question interpretation is addressed as the automatic identification of the set of relevant relations between entities in the natural language input question, matched against a repository of automatically collected relational patterns (i.e. the WikiFramework repository). Such patterns represent possible lexicalizations of ontological relations, and are associated to a SPARQL query derived from the linked data relational patterns. Wikipedia is used as the source of free text for the automatic extraction of the relational patterns, and DBpedia as the linked data resource to provide relational patterns and to be queried using a natural language interface.

Keywords: Question Answering, Linked Data, relation extraction

1 Introduction

As the social web is spreading among people and the web of data continues to grow (e.g. Linked Data initiatives), there is an increasing need to allow easy interactions between non-expert users and data available on the Web. In this perspective we address the development of methods for a flexible mapping between natural language expressions, and concepts and relations in structured knowledge bases. Specifically, we present QAKiS, the Question Answering system that we have submitted at the QALD-2 evaluation exercise.

QAKiS, Question Answering wiKiframework-based System, allows end users to submit a query to an RDF triple store in English and get results in the same language, masking the complexity of SPARQL queries and RDFS/OWL inferences involved in

the resolution, while profiting from the expressive power of these standards. In the actual implementation, QAKiS addresses the task of QA over structured knowledge bases (e.g. DBpedia), extracted from corpora in natural language. QAKiS builds on top of previous experiences on QA over structured data in closed domains, particularly the QALL-ME system [3], aiming at enhancing the scalability of the approach and its portability across domains. A crucial issue in Question Answering over linked data, and a major focus of the QAKiS system, is the interpretation of the question in order to convert it into a corresponding query in a formal language (e.g. SPARQL). Most of current approaches (e.g. PowerAqua [4]) base this conversion on some form of flexible matching between words of the question and names of concepts and relations of a triple store. However, a word-based match may fail to detect the relevant context around a word, without which the match might be wrong. In our first participation at QALD-2 we try to exploit a relation-based match, where fragments of the question are matched to relational textual patterns automatically collected from Wikipedia (i.e. the WikiFramework repository). The underlying intuition is that a relation-based matching would provide more precision w.r.t. matching on single tokens, as done by current QA systems on linked data.

2 WikiFramework patterns

Our QA approach makes use of relational patterns automatically extracted from Wikipedia and collected in the WikiFramework repository [5]. Relational patterns capture different ways to express a certain relation in a given language. For instance, the relation `birthDate(Person, Date)` can be expressed in English by the following relational patterns: [`Person` was born in `Date`], [`Person`, (`Date`)], [`Person`, whose date of birth is `Date`].

Goal of the WikiFramework is to establish a robust methodology to collect relational patterns in several languages, for the relations defined in DBpedia ontology (similarly to [2,6]). Therefore, following the semantic web RDF model, each relation is represented by a set of triples $\langle S, P, O\rangle$ where S is the subject (domain) of the relation, O is the object (range), and P (predicate or property) is the name of the relation. For example, an instance of the `crosses` relation is:

```
<http://dbpedia.org/resource/Golden_Gate_Bridge>
<http://dbpedia.org/ontology/crosses>
<http://dbpedia.org/resource/Golden_Gate>
```

or using namespaces and prefixes:

```
dbr:Golden_Gate_Bridge dbo:crosses dbr:Golden_Gate
```

We assume that there is a high probability that the structured information present in the Infobox (i.e. a table put in the right-hand corner of most of Wikipedia articles providing a structured summary of information mentioned in the text) is also expressed using natural language sentences in the same Wikipedia page. Therefore as a first step, we collect all the triples for the 1296 DBpedia ontology relations: for each of them we extract the subject (e.g. `Golden Gate Bridge`), and we automatically match each DBpedia relation with the

Wikipedia pages whose topic is the same as the subject, and in which such relation is reported in the Infobox. For instance, given the relation `crosses` reported above, the Wikipedia page about the Golden Gate Bridge is selected (Figure 1).

As a second step, we collect all the sentences in the selected Wikipedia pages where both the strings of subject and object of the relation match. For instance, in the page about the Golden Gate Bridge (Figure 1), the sentence "The Golden Gate Bridge is a suspension bridge spanning the Golden Gate" is detected, since both entities match (i.e. domain: Golden Gate Bridge, range: Golden Gate). Such sentence is extracted and the subject and object are substituted with the corresponding DBpedia ontology classes (i.e. for the relation `crosses`, `Bridge` is the domain and `River` is the range). The pattern [The `Bridge` is a suspension bridge spanning the `River`] is obtained and stored in a pattern repository.

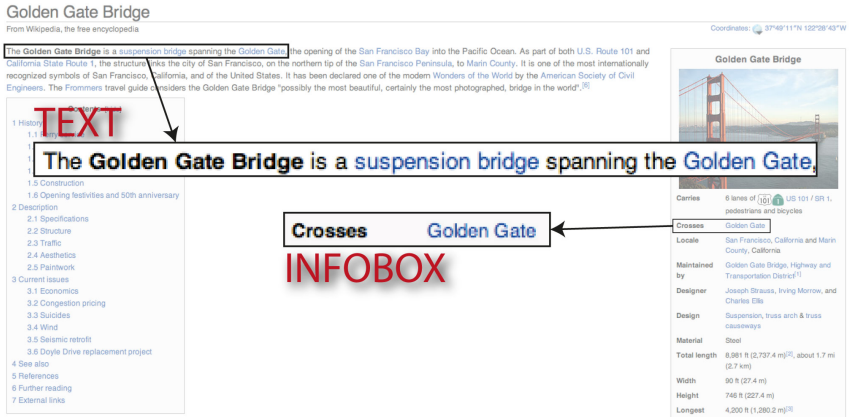

Fig. 1. An example of Wikipedia page with infobox.

To increase the recall of the pattern extraction algorithm outlined above, we apply different matching strategies. The first one involves the exact match of the entire string (the subject), as provided by Wikipedia. If the algorithm does not find such string, we clean it deleting the expressions between brackets, used to disambiguate pages with the same title, as in "Carrie (novel)" and "Carrie (1976 film)". In the final step, the original string is tokenized and lemmatized using Stanford CoreNLP[4] and, given the set of obtained tokens, a new string is built by combining them (preserving the original word order). For instance, starting from "John Fitzgerald Kennedy", we obtain the new strings "John Fitzgerald", "John Kennedy", "Fitzgerald Kennedy", "John", "Fitzgerald" and "Kennedy". In case of numeric or time ranges, we apply the entity extractor of Stanford CoreNLP, and match the value in the extracted string with the given normalized value.

[4] `http://nlp.stanford.edu/software/corenlp.shtml`

Once the patterns for all relations are collected, we cluster them according to the lemmas that are present between the domain and the range. Then, we sort such patterns according to the frequency they appeared in Wikipedia pages, and we wipe out: *i)* the ones whose frequency is less than 2, and *ii)* the ones that contain only non discriminative words (e.g. punctuation marks, prepositions, articles, etc. as in the pattern [`Person` (`Date`)]). Table 1 reports an extract of the set of patterns collected for the relations `spouse` and `crosses`.

Table 1. Examples of patterns.

Relation	Patterns
`spouse`	`Person` wife `Person`
	`Person` married `Person`
	`Person` husband `Person`
`crosses`	`Bridge` spanning `River`
	`Bridge` bridge `River`
	`Bridge` crossing `River`

As the last step, three sets of keywords for each relation are created, respectively for most frequent tokens, lemmas and stems. Each set contains 20 words, sorted by the frequency of presence in the collected patterns. To further improve recall, we finally append to the sets of keywords the tokens, lemmas and stems extracted from the CamelCase name of the relation (e.g. the tokens "birth" and "date" are added to the keywords of the relation `birthDate`).

3 QAKiS system description

This section presents the architecture of QAKiS, and a step by step description of the system work-flow.

3.1 System architecture

QAKiS is composed of two main components (as shown in Figure 2):

- the *query generator* is the entry point to the system: it reads the questions from the input file, generates the typed questions to be matched with the patterns (as explained in Section 3.2), and then generates the SPARQL queries from the retrieved patterns (Section 3.3). If several SPARQL queries are generated, the query with the highest matching score is selected among those that have at least one result;
- the *pattern matcher* takes as input a typed question, and retrieves the patterns (among those stored in the pattern repository) that match such question with the highest similarity (Section 3.2).

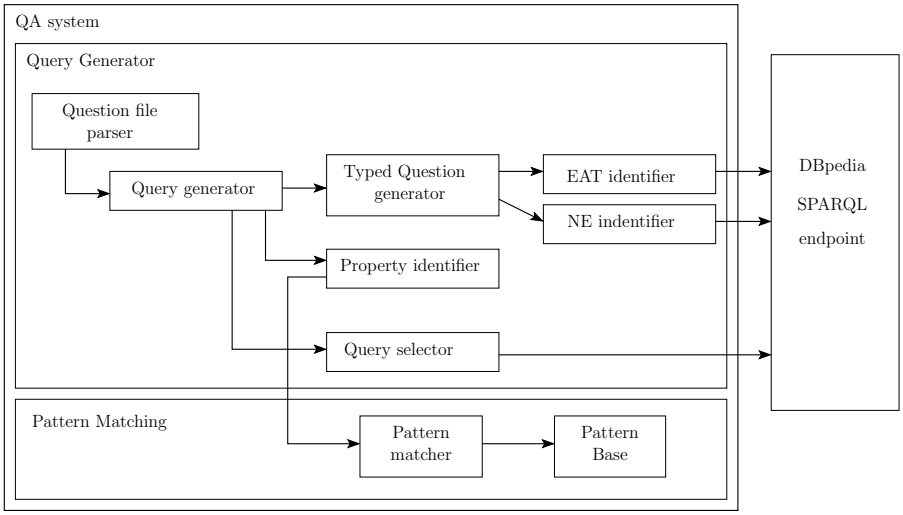

Fig. 2. QAKiS architecture.

3.2 Typed questions generation and pattern matching

The first version of QAKiS (with which we participated in the QALD-2 challenge) targets questions containing a Named Entity (NE) that is related to the answer through one property of the ontology, as *Which river does the Brooklyn Bridge cross?* (id=4, training set). According to our WikiFramework-based approach, such questions match a single pattern (i.e. one relation). Before running the pattern matcher component, we replace *i)* the NE present in the question by its types, and *ii)* the question keywords by the Expected Answer Type (EAT), obtaining what we call a *typed question* (e.g. `[River]` does the `[Bridge]` cross?). The answer can then be retrieved with a SPARQL query over a single triple. In the following, each step of the system work-flow is explained in details.

EAT identification To implement a generic method for QA, we do not take advantage of the attribute *answertype* provided by the QALD-2 organizers for each question, to detect the EAT. Instead we implemented simple heuristics to infer it from the question keyword, i.e. if the question starts with:

- "When", the EAT is `[Date]` or `[Time]`;
- "Who", the EAT is `[Person]` or `[Organisation]`;
- "Where", the EAT is `[Place]`;
- "How many", the EAT is `[nonNegativeNumber]`;
- either "Which", "Give me all", "List all", we expect the EAT to be the terms following the question keyword (we consider the longest term matching a class label in the DBpedia ontology).

NE identification Three strategies are followed, applied in a cascade order (i.e. if a strategy fails, we try the next one):

1. we run the Named Entity Recognizer in Stanford Core NLP to identify the NEs. If the question contains the label of an instance from the DBpedia ontology, and this label contains one of the NEs identified by Stanford, then we keep the longest expression containing it. This is usually the best option, however we noticed that for many expressions containing "of" (e.g. *mayor of Berlin*), it was better to keep only what comes after "of" (i.e. *Berlin*), otherwise the page "Mayor of Berlin" (present in Wikipedia) is selected. In such cases, we extract both NEs;
2. if Stanford does not recognize any NE, we look if it recognizes at least one proper noun (whose Part of Speech is NNP). If this is the case, we take the longest label in the DBpedia ontology containing it;
3. if the two first strategies fail, we just look for the longest DBpedia instance label found in the question.

Typed questions generation Once the set of EATs and NEs are identified for a question, the typed questions are generated by replacing the question keywords by the supertypes of the EAT, and the NE by its types. For instance, for the question "Who is the husband of Amanda Palmer?" (id=42, test set), as EAT both `[Person]` or `[Organisation]` (both of them subclasses of `[owl:Thing]`) are considered. Moreover, the NE *Amanda Palmer* has type `[MusicalArtist]`, `[Artist]` and `[owl:Thing]`. All 9 typed questions are generated.

WikiFramework pattern matching Before pattern matching, stopwords in the typed questions are deleted, and stems, lemmas and tokens are extracted. A Word Overlap algorithm is then applied to compare the stems, lemmas, and tokens of questions with the patterns and keywords for each relation. A similarity score is provided for each match: the highest represents the most likely relation.

3.3 Generation of SPARQL queries

A set of patterns is retrieved by the pattern matcher component for each typed question (we restrict the number of patterns retrieved to 5 per typed question). The patterns retrieved for all the typed questions are sorted by decreasing matching score. For each of them, one or two SPARQL queries are generated, either *i)* `select ?s where{?s <property> <NE>}`, *ii)* `select ?s where{<NE> <property> ?s}` or *iii)* both, according to the compatibility between their types and the property domain and range. Such queries are then sent to the SPARQL endpoint. As soon as a query gets results, we return it; otherwise we try with the next pattern, until a satisfactory query is found or no more patterns are retrieved.

4 Results on QALD-2 data

In this section we present and discuss QAKiS' performances on extracting correct answers for natural language questions (provided by the QALD-2 challenge) from DBpedia. Given a certain question, QAKiS carries out a fully automatic process (Section 3), and as output specifies both a SPARQL query and the answers, as obtained querying the SPARQL endpoint provided by the challenge organizers.[5] Table 2 reports on the evaluation both on QALD-2 training and test set, with respect to precision, recall and f-measure. Furthermore, statistics are provided with respect to the number of questions for which QAKiS found an answer, and among them, the number of correct and partially correct answers.

Table 2. QAKiS performances on DBpedia data sets

	DBpedia	
	Training set	**Test set**
Precision	0.476	0.39
Recall	0.479	0.37
F-measure	0.477	0.38
# answered questions	40/100	35/100
# right answers	17/40	11/35
# partially right answers	4/40	4/35

4.1 Error analysis and discussion

As introduced before, the current version of QAKiS addresses questions expressing only one relation between the subject (the entity in the question) and the object of the relation (the Expected Answer Type). More precisely, for now the subject must be a NE (that QAKiS recognizes as described in Section 3.2). Even if the strategies we defined for NER seem to correctly recognize most of the NEs present in the questions, a module for coreference resolution should be added to capture e.g. the DBpedia entry *Juliana_of_the_Netherlands* from the question: *In which city was the former Dutch queen Juliana buried?* (id=89, test set). Moreover, the strategy we added to deal with cases such as *the mayor of Berlin*, i.e. consider as NE both the one recognized by Stanford NER and the longest instance of the DBpedia ontology containing it, fails to correctly capture the book title in *Who wrote the book The pillars of the Earth?*, since patterns that match with a higher score are found for *Earth* (the NE recognized by Stanford).

Most of QAKiS' mistakes concern wrong pattern matching, i.e. the highest similarity between a typed question (generated as described in Section 3.2) and patterns in the pattern repository, is provided for patterns expressing the wrong

[5] `http://greententacle.techfak.uni-bielefeld.de:5171/sparql`

relation. As described in Section 3, such similarity is calculated using a Word Overlap algorithm. We plan to substitute such algorithm with more sophisticated approaches, to consider also the syntactic structure of the question.

Another problem we faced concerns the fact that patterns (and questions) can be ambiguous, i.e. two questions using the same surface forms can in fact refer to different relations in the DBpedia ontology (e.g. *Who is the owner of Universal Studios?* relies on the relation `owner` for answer retrieval, while in *Who owns Aldi?* the correct answer is the subject of the relation `keyPerson`). A possible solution could be clustering similar relations (with several patterns in common), so that the system tries to find an answer for all the relations in the cluster, and selects the relation providing a non null answer.

Considering the ontology we rely on for relation extraction and answer retrieval, QAKiS addresses only the questions tagged by QALD-2 organizers as `onlydbo:true` (i.e. where the query relies solely on concepts from the DBpedia ontology). Surprisingly, in a few cases we were also able to provide correct answers for questions tagged `onlydbo:false`, e.g. in *What is the time zone of Salt Lake City?* (id=58 test set), matching the relation `timeZone`, or in *Give me all video games published by Mean Hamster Software* (id=71, training set), matching the relation `publisher` (since such company develops only video games).

Most of the partially correct answers we provide concern questions considering more than one relation, but for which QAKiS detects only one of them (due to the actual version of the algorithm). For instance, for *Give me all people that were born in Vienna and died in Berlin.* (id=19, test set), we retrieve and list all the people that died in Berlin. We plan to target such kind of questions in a short time, since the two relations are easily separable. On the contrary, more complex strategies should be thought to deal with questions with a more complex syntactical structure, as *Who is the daughter of Bill Clinton married to?* (id=3, training set), for which at the moment we answer with the name of Bill Clinton's wife (we match only the relation `spouse`).

We plan short term solutions for boolean questions as *Is the wife of president Obama called Michelle?* (id=7, training set): we correctly match the relation `spouse` but we provide as answer *Michelle_Obama*, instead of the boolean *true*.

5 Conclusions

This paper describes our first participation to the QALD-2 open challenge using the QAKiS system, a QA system over linked data (in particular, DBpedia) based on the WikiFramework approach. In general, the results we obtained using the proposed approach are in line with other systems' results, fostering future research in this direction. Due to the high variability and complexity of the task, much work is still to be done, and improvements should be planned on different fronts: *i)* extending the WikiFramework pattern extraction algorithm following [6], *ii)* improving the NE detection strategy, and addressing also questions whose subject is a class and not a specific NE (e.g. *What is the longest river?*(id=15, test set); *iii)* investigating the applicability of the Textual En-

tailment approach (a framework for applied semantics, where linguistic objects are mapped by means of semantic inferences at a textual level [1]) to improve the question-pattern matching algorithm; *iv)* addressing boolean and questions requiring more than one relation, to increase the system coverage. Finally, future work will also address the issue of natural language answer generation.

References

1. Dagan, I., Dolan, B., Magnini, B., Roth, D. (2009), Recognizing textual entailment: Rational, evaluation and approaches, in JNLE vol. 15 (4), pp. i-xvii.
2. Gerber, D., Ngonga Ngomo, A.C. (2011), Bootstrapping the Linked Data Web, in 1st Workshop on Web Scale Knowledge Extraction ISWC 2011, Bonn, Germany.
3. Ferrandez, O., Spurk, C., Kouylekov, M., Dornescu, I., Ferrandez, S., Negri, M., Izquierdo, R., Tomas, D., Orasan, C., Neumann, G., Magnini, B., Vicedo, J.L. (2011), The QALL-ME Framework: A specifiable-domain multilingual Question Answering architecture, in Web Semantics: Science, Services and Agents on the World Wide Web journal, vol. 9 (2), pp. 137-145.
4. Lopez V., Uren V., Sabou, M., Motta, E. (2011), Is Question Answering fit for the Semantic Web?: a Survey, in Semantic Web - Interoperability, Usability, Applicability journal, vol. 2(2), pp. 125-155.
5. Mahendra, R., Wanzare, L., Bernardi, R., Lavelli, A., Magnini, B. (2011), Acquiring Relational Patterns from Wikipedia: A Case Study, in Proceedings of the 5th Language and Technology Conference, Poznan, Poland.
6. Wu F., Weld, D.S. (2010), Open information extraction using Wikipedia, in Proceedings of ACL, Uppsala, Sweden.