Intui2: A Prototype System for Question Answering over Linked Data

Corina Dima

Seminar für Sprachwissenschaft, University of Tübingen,
Wilhemstr. 19, 72074 Tübingen, Germany
corina.dima@uni-tuebingen.de

Abstract. An ever increasing amount of Linked Data is made available every day. Public triple stores offer the possibility of querying hundreds of millions of triples. But this information can only be retrieved using specialized query languages like SPARQL, so for the majority of Internet users, it is still unavailable. This paper presents a prototype system aimed at streamlining the access to the information stored as RDF. The system takes as input a natural language question formulated in English and generates an equivalent SPARQL query. The mapping is based on the analysis of the syntactic patterns present in the input question. In the initial evaluation results, against the 99 questions in the QALD-3 DBpedia test set, the system provides a correct answer to 30 questions and a partial answer for another 3 questions, achieving an F-measure of 0.32.

Keywords: Question Answering, Linked Data, Semantic Search

1 Introduction

The rise of semantic web technologies in the last decade has had an important outcome: large amounts of structured data have been made available in standard formats such as RDF and OWL. Big data repositories fostered the development of efficient algorithms for storing, indexing and querying RDF.

Collaborative knowledge bases such as DBpedia [2] and Yago2 [8] now contain millions of interlinked facts extracted with high accuracy from manually created structured data, like Wikipedia, GeoNames and WordNet. The access to all this information is, however, restricted. Specialized query languages, such as SPARQL, are the only interfaces available. For this reason, finding information in RDF stores presupposes, on one hand, being comfortable with the store's query language, and on the other hand, having a good understanding of how the data was modelled.

Consider, for example, Q71 from the QALD-3 test set, *When was the Statue of Liberty built?* The SPARQL query that would provide the correct answer is shown below. As you can see, there is no string similarity between the property that provides the correct answer, *dbp:beginningDate*[1], and the verb used in the question, *built*. Additional knowledge about how buildings are modelled in DBpedia, specifically how the building date is defined, is necessary in order to select the correct predicate.

SPARQL query for Q71 from the QALD-3 test set, "When was the Statue of Liberty built?"

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX res: <http://dbpedia.org/resource/>
SELECT DISTINCT ?date
WHERE {
    res:Statue_of_Liberty dbp:beginningDate ?date .
}
```

The need for more advanced, semantic, question answering systems that operate over large repositories of Linked Data has also been the motivation for the QALD (Question Answering over Linked Data) series of workshops. The first workshop, QALD-1 [12], was organized in 2011 and it offered a test set containing 100 questions whose answer could be found in the DBpedia and MusicBrainz [13] datasets (50 each). QALD-2 and QALD-3 offered 200 questions over the same datasets (100 each). The questions vary in length and complexity and are meant to be indicative of what a typical user of such a semantic question answering system would ask.

This paper presents Intui2, a prototype for an RDF backed question answering system that can transform transform natural language questions into SPARQL queries, thus giving the end users access to the information stored in RDF repositories. The name of the system comes from the Romanian verb *a intui* (Engl. *to intuit*), which means *"to know, sense or understand by intuition"*[2].

The system is based on three central ideas:

(1) Grammatically correct questions are built of *synfragments*. A *synfragment* corresponds, from a syntactic point of view, to a subtree of the syntactic parse tree of the question. From a semantic point of view, a *synfragment* is a *minimal* span of text that can be interpreted as a concept URI, as an RDF triple or as a complex RDF query (see Fig. 1 for an example).
(2) The interpretation of a *parent synfragment* is obtained by combining the interpretations of its child synfragments in a way that is particular to the parent synfragment due to its syntactic and/or semantic characteristics (see end of section 2.2 for details).
(3) The interpretation of a question in an RDF query language can be composed though a recursive interpretation of its synfragments, visited in a *most-informative-first* order (see section 2.2 for details).

[1] *dbp* denotes the DBpedia property namespace, *http://dbpedia.org/property/*,*dbo* the DBpedia ontology namespace, *http://dbpedia.org/ontology/* and *res* the DBpedia resource namespace, *http://dbpedia.org/resource/*

[2] according to the definition in the Merriam-Webster online dictionary, http://www.merriam-webster.com/

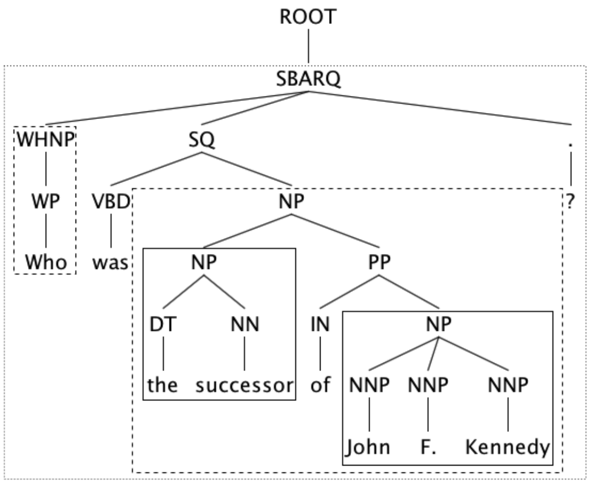

Fig. 1. *Synfragments* for Q2 from the QALD-3 DBpedia test set, *Who was the successor of John F. Kennedy?* Solid boxes surround synfragments that are interpreted as URIs, dashed boxes mark synfragments that are interpreted as one triple and dotted boxes indicate synfragments that correspond to a complex SPARQL query.

The system interprets the question with respect to the provided RDF backend, by mapping the occurring natural language expressions to concepts in the triple store.

2 System Description

The system receives as input a natural language question formulated in English and outputs the query that will retrieve the answer to the question from the RDF backend. The architecture of the system is illustrated in Fig. 2.

2.1 Preprocessing Phase

The natural language question is first preprocessed using the Stanford CoreNLP suite ([11], [14]), in particular tokenized, lemmatised, POS tagged and parsed. An additional preprocessing step is retrieving the *information rank (IR)* for each token. The IR gives an estimation of the specificity of a token: frequently occurring words, such as determiners, pronouns, verbs like *is, has, does, give, married* or nouns such as *country, river, city* get a small IR, while less common words like *monarchical, astronauts* or proper names are assigned a high IR. To compute the information rank we constructed a large word list by taking all the words in a Wikipedia dump, converting them to lower case, computing their frequencies over the whole Wikipedia and then sorting them in decreasing order of frequency. The *information rank* of a word is the index of the word in this sorted word list. The list has about 880,000 entries, and does not contain the words that appear less than 10 times in the whole Wikipedia. If a word is not in the list, it is assigned the rank -1. To ensure good coverage of the DBpedia concepts in the QALD-3 test set, we generated the word list using the same Wikipedia dump used to extract DBpedia 3.8.

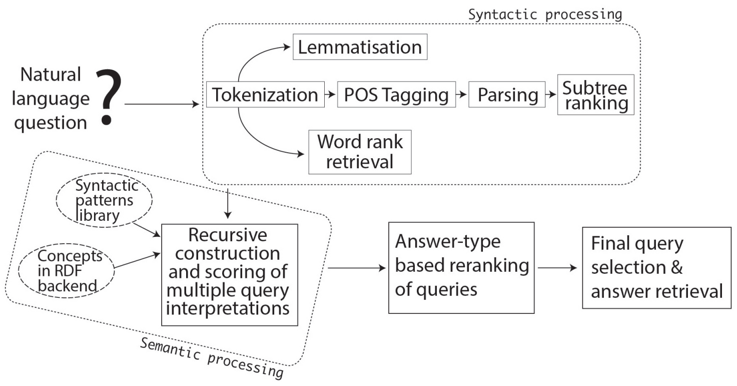
Syntactic processing
Natural language question
Lemmatisation
Tokenization
POS Tagging
Parsing
Subtree ranking
Word rank retrieval
Syntactic patterns library
Concepts in RDF backend
Recursive construction and scoring of multiple query interpretations
Answer-type based reranking of queries
Final query selection & answer retrieval
Semantic processing

Fig. 2. Intui2 system architecture.

2.2 Analysis Phase

All the data gathered in the preprocessing phase is provided as input for the analysis phase. The analysis of a question entails the recursive traversal of the question's parse tree, starting at the ROOT node, in a *most-informative-first* order. The order is established by computing the *cumulative rank (CR)* of each subtree of the current node. The *cumulative rank* of a tree is the sum of the *information ranks* of all its leaf nodes. The subtrees of each node are then traversed in decreasing cumulative rank order. In the case of the parse tree in Fig. 3, the nodes are processed in the following order: first the NP over *"the same timezone"* with the highest CR, 135.633, then the NP over *"Utah"* with the CR 3.303, than the IN over *"as"*, with the CR 16, then the parent PP, with the CR 3.319, than the NP with CR 138.952, the IN with CR 6, the PP with CR 138.958 and so on until the whole tree has been processed.

The system distinguishes between two types of nodes: those that correspond to low-level syntactic patterns (preterminals and pre-preterminals) and those corresponding to high-level syntactic patterns (non-terminal nodes that are neither preterminals nor pre-preterminals). The syntactic pattern of a node is the label of the node plus the labels of all its immediate children, in a left-to-right order. For example, in Fig. 3, the syntactic pattern corresponding to the NP node above *"the same timezone"* is NP DT JJ NN. The difference between the two levels is that a node with a low-level syntactic pattern is always analysed as

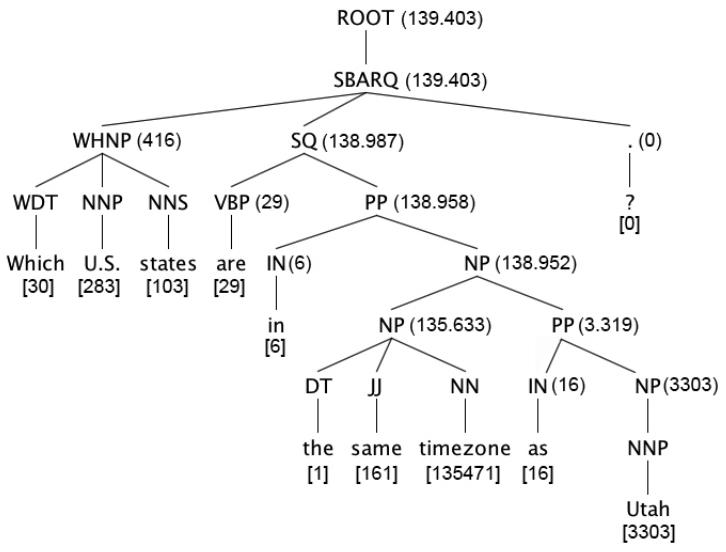

Fig. 3. Information ranks (enclosed by square brackets) and cumulative ranks (enclosed by parentheses). Punctuation marks are assigned the information rank 0.

a URI whereas for the high-level syntactic patterns the system must construct the interpretation of the node's subtrees before the actual interpretation of the node.

The syntactic patterns for the QALD-3 DBpedia test set were obtained by parsing the questions and then doing a recursive traversal of each of the parse trees. During the traversal, the syntactic pattern of each non-terminal node was added to the global set of patterns. We obtained this way 138 syntactic patterns, split into 16 groups based on the syntactic category of the head node: 52 patterns that start with an NP node, 26 patterns that start with a VP node, 13 patterns with a WHNP node, etc. As an exercise, we computed the number of syntactic patterns in the QALD-3 DBpedia training set, and we discovered 154 syntactic patterns grouped in 19 categories. To measure the overlap between the two, we counted the number of patterns in the combined test and training set, which resulted in 204 syntactic patterns grouped in 19 categories. This shows that while the number of possible syntactic patterns is virtually infinite, in practice a relatively small number of syntactic patterns can cover a large amount of natural language questions similar to the ones in the QALD datasets.

For each syntactic pattern we then manually provided a mapping suggestion. Due to the larger number of patterns that had to be mapped, we only provided mapping suggestions for the syntactic patterns covering the QALD-3 DBpedia test set. A mapping suggestion specifies two attributes:

(i) The slot that this syntactic pattern should fill in the triple: either subject, object or predicate. In ambiguous cases, the pattern can used to fill two or even all three slots.
(ii) The URI pattern: for example, predicate URI patterns are built through concatenation using the camel-case convention, while the subject and object URI patterns are formed by concatenation using the underscore as a delimiter.

For example, one of the most frequent syntactic patterns is NP NNP NNP, that is a noun phrase made of two proper nouns, like *Benjamin Franklin*. The mapping suggestion provided by the system for this pattern is: (slot - subject or object, URI pattern - NNP_NNP). Another frequent pattern is NP DT NN, a noun phrase made from a determiner and a common noun. For this pattern the system generates a mapping suggestion covering all three slots (subject, predicate, object). The URI pattern for the subject and object slots is the capitalized version of the NN, and the URI pattern for the predicate slot is the NN itself. So for example *the capital* has the URI pattern *Capital* when in the subject or object slot and the URI pattern *capital* when in the predicate slot. Observe that both URI patterns ignore the determiner, which proved uninformative in the case of this syntactic pattern.

The analysis begins with an empty queue of analysis results. At this point, the current node can only be analysed as a single URI, depending on the mapping suggestion for the current syntactic pattern. The result of this step is one or more lists of URIs, depending on how many slots were defined by the mapping. This result is added to the queue of analysis results and the analysis is continued with the next node.

When the analysis queue is non-empty, the analysis can proceed along one of the following two paths:

- **The top of the queue is of type URI.** In this case, the result that is the head of the queue is popped out and the current node is analysed with respect to it. If the result specifies a subject or object URI, then the current node is interpreted as a predicate. If the result specifies a predicate URI, then the current node is first interpreted as a subject/object URI and then the existing result predicate is reinterpreted with respect to it.
  The interpretation of a predicate given a subject or object URI involves querying the triple store for all the triples with that subject or object and then scoring each of the predicates obtained with respect to the specified predicate pattern.
  If at the end of this analysis there is no resulting query, the initial URI result is re-added to the queue, with a flag that specifies that it has already been analysed once. This allows the system to give the answer "OUT OF SCOPE" when no corresponding predicate is found, which means that the answer to a specific question is not covered by the data in the current RDF backend.
- **The top of the queue is of type Query.** In this case, the top result is removed from the queue and it is rewritten by transforming any occurrence

of the answer variable into a variable with a random name, e.g. *randomVar*. Then the system generates two queries: one where the *randomVar* is the subject, and one where *randomVar* is the object of the query. The queries are then scored and added to the queue of analysis results, and the analysis continues.

2.3 Scoring Synfragments

Every synfragment is assigned a score between 0 and 1. Complex synfragments are scored by multiplying the scores of all their constituents. URI synfragments are scored in the following manner:

- **Subject/object URIs** are scored using a simple, bigram-based string similarity measure between the words in the question and the concepts in the RDF backend.
- **Predicate URIs** are scored initially using the same string similarity measure between the surface form of the predicate and the properties in the backend. If the score is under a specified threshold (0.5), then the predicate is re-scored using the WordNet similarity measure described by Hirst and St. Onge (HSO) [7] and implemented in WS4J [16]. For example, the pair *(nicknames, http://dbpedia.org/property/nickname)* has a string similarity score of 0.93, whereas the pair *(mayor, http://dbpedia.org/ontology/leader)* has a string similarity score of 0, but a HSO score of 0.375.

Additionally, the URIs that belong to the *http://dbpedia.org/ontology/* namespace are boosted by doubling the initially assigned score. This makes the system favour the URIs from the manually corrected DBpedia ontology as opposed to those that are part of the automatically extracted *http://dbpedia.org/property/* namespace.

2.4 Reranking Phase

All the possible partial analyses of a question are scored and kept until all the synfragments of the question are analysed. Given the syntactic patterns present in the question, the expected answer type can sometimes be very easily inferred. For example, questions that begin with *Who...* expect an answer of type *dbo:agent* or *dbo:person*. This can be easily specified in the query by means of the *rdf:type* property in the following manner:

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
SELECT DISTINCT ?answer
WHERE {
    ?answer rdf:type dbo:Person .
}
```

The *correctness* of a given query is then measured as the number of results that have the correct answer type divided by the total number of answers. The final score of the query is obtained by multiplying the existing score with the computed correctness. This technique provides us with a very precise reranking of the end results, but unfortunately not all the questions have an easy to extract pattern when it comes to the expected answer type. The current system only has rerankers for the case when the answer should be an *Agent/Person* and for when the answer should be a number.

3 System Evaluation

The QALD-3 challenge offered 99 test questions whose answer had to be found either in DBpedia or in a federated triple store containing additional data from Yago2 and MusicBrainz. In terms of complexity, the test questions are either simple questions, whose interpretation only involves recognizing the components of a triple, or complex questions that require aggregation or some form of semantic resolution to be interpreted.

Our prototype system currently focuses on the simple questions, as it is not yet equipped for dealing with complex natural language constructions. This reduces the number of questions our system could actually have constructed a correct query for to 67. Our system got 30 of the questions right, and for another 3 it got a partially right answer. Out of these, 26 questions could be answered via a query with a single triple, 3 with a query with two triples and 4 had the answer OUT OF SCOPE. The results over the QALD-3 test and training sets are detailed in Table 1. The difference in the F-measure between the training and the test set can be explained by the fact that the train set was run against the same system as the test set, without manual addition of mapping suggestions for the syntactic patterns that occurred only in the training set. The system, running on a single core 2.4Ghz Intel processor with 4GB of RAM, needed on average 103 seconds to answer a question. The required time depends directly on the complexity of the question itself (how many triples have to be constructed) and on the concepts that appear in it (a question containing the name of a country or continent will take longer to process because there are usually many triples referring to such a concept, while processing a proper name will usually involve the analysis of much fewer triples).

For efficiency reasons, we made use of a locally installed version of DBpedia, powered by Jena TDB [10]. The repository can be queried using the Jena ARQ query engine [9]. Given that the triple store contains over 150 million triples, doing string matching queries directly is very ineffective and time consuming. To mitigate this problem we generated 3 index files, containing all the unique subjects, predicates and objects of the triples in the repository. We used these index files to speed up the lookup times for URIs. The additional markup existent in DBpedia is used to ensure a better mapping from concepts to their lexicalisations. The system searches for triples with the predicate *dbo:wikiPageRedirects* for all the URIs retrieved via the index. If such triples are found, they are then added to the list of candidate URIs for the specified pattern.

Table 1. Evaluation results for the Intui2 system

Test Set	Total	Right	Partially	Recall	Precision	F-measure
QALD-3 DBpedia test	99	30	3	0.32	0.32	0.32
QALD-3 DBpedia train	100	18	2	0.2	0.19	0.19

3.1 Problematic Cases

We analysed the questions where our system gave a wrong answer. These questions can be split into the following categories:

- **Synfragments that should be interpreted directly as a query.** We discovered synfragments that correspond to another interpretation paradigm: the answer is an entity of a specified type with a specified property. For example, *books by Kerouac*, from Q81, should not be interpreted by looking at the concept *Kerouac* and then searching for triples with a predicate like *books*, but rather directly using the query

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
SELECT DISTINCT ?answer
WHERE {
    ?answer rdf:type dbo:Book .
    ?answer dbo:author res:Jack_Kerouac .
}
```

  The system could be improved by constructing multiple possible interpretations of a question, e.g. by interpreting upfront such a synfragment using both the general triple matching and the more specialised type matching paradigms. Other synfragments that display the same problem are are *lakes in Denmark, Argentine films, companies in Munich,* etc.
- **Cases when both the string similarity and the HSO scorer failed to assign a high score to the correct predicate**. The actual reasons for this are quite diverse, but this is obviously one of the biggest challenges for such a system. An RDF predicate can be verbalized in tens of different ways in natural language, so finding a mapping between the two is not always trivial. The predicate used to model the relation can be lexically and semantically different from the verbalisation used in the question, as is the case with the pair *(beginningDate, built)* from Q71 that was described in the Introduction. The WordNet similarity measures also fail when the compared items have separate POS tags, but clearly belong to the same semantic field, like the pair *(die, death)* from Q74, or when the words are not in the WordNet database,

like in the example *(placeOfBurial, buried)*, where *burial* is not covered by WordNet. Moreover, there are cases when the DBpedia predicate maps to complex natural language expressions, like in the case of Q98, *Where does the creator of Miffy come from?*, where the pattern *Where does X come from?* should be mapped to the predicate *dbo:nationality*.
Solving these cases would imply finding new ways of mapping predicates to natural language expressions. Promising methods include concept embeddings [3], Extended Semantic Analysis (ESA) [5] and using automatically extracted pattern libraries such as BOA [6].

- **Predicates that have a given property both as a subject and as an object**, like *dbo:parent*. For Q67, *Who are the parents of the wife of Juan Carlos I?*, *the Queen Sofia of Spain* is the object of the relation *dbo:parent* whose subject are her parents and the subject of the same relation when the object are her own children. The system generates both interpretations and gives them both the score 1, and chooses randomly one of them at the end of the analysis, in this case the wrong one.
For choosing the correct triple in this case, the system would have to offer a more precise specification of the predicate, for example by adding an extra test triple with the inverse relation.
- **Named Entity Recognition** In Q97, *Who painted The Storm on the Sea of Galilee?*, the system fails to recognize that *The Storm on the Sea of Galilee* is the name of a famous painting by Rembrandt and tries to interpret *Galilee* as a location and then *the sea* and *the storm* as predicates relating to it.
Such errors can be avoided through an extra preprocessing step for identifying such named entities with the help of large indexes of proper names (which could be directly extracted from Wikipedia).

4 Existing Approaches

Several systems were already tested in the context of the QALD workshops, all presenting different approaches to the interpretation of natural language questions as SPARQL queries. The system of Unger et. al [15] operates under the assumption that the template of the target SPARQL query can be determined by analysing the syntactic structure of the question and by interpreting it with respect to an ontology-based grammar. The constructed template contains empty slots, that have to be further specified in a second step of the analysis, that deals with entity identification and property detection. Their system, evaluated on the QALD-1 test set of 50 DBpedia-related question, answered 19 questions right and got a partial answer for another 2 questions.

Aggarwal et. al [1] describe an approach that uses typed dependency information to guide the identification of DBpedia concepts. The predicates are matched using relatedness measures based on WordNet. The evaluation against the QALD-2 test set of 100 DBpedia questions revealed that the system could answer 32 questions right and had a partially correct answer for an extra 7 questions.

The QAKiS system presented by Cabrio et. al [4] introduces yet another method of interpreting natural language questions as SPARQL queries. Their approach is based on the automatic identification of a set of relevant relations between entities in the natural language question and their subsequent matching against a repository of relational patterns automatically extracted from Wikipedia. The system answers 11 questions correctly and is partially right in the case of 4 questions from the 100 test questions in the QALD-2 challenge.

5 Conclusion and Future Work

We have presented a prototype system for question answering over Linked Data, that can answer natural language questions with respect to a given RDF backend by analysing them in terms of the synfragments they are composed of.

We plan to further improve the system, first by redesigning the synfragment interpretation paradigm to accommodate the interpretation of the same synfragment on multiple levels (as a URI, triple or query). Also, we plan to look into more advanced techniques for mapping natural language input to database concepts, like concept embeddings and Explicit Semantic Analysis.

The mapping of the syntactic patterns to slots and URI patterns was done manually for this version of the system. We intend to enhance the system with an automatically induced mapping, obtained through the exploration of large amounts of syntactic patterns.

Our intention is to extend the system to other input languages, either directly, by integrating a machine translation module that would translate the initial queries to English, or by integrating language processing tools for other languages and then mapping directly to the DBpedia concepts from those languages.

Acknowledgements The research leading to these results has received funding from the European Commission's 7th Framework Program under grant agreement no. 238405 (CLARA).

References

1. Aggarwal, N., Buitelaar, P.: A System Description of Natural Language Query over DBpedia. In: Proceedings of Interacting with Linked Data, pp. 96–99. Heraklion, Greece (2012)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia – A Crystallization Point for the Web of Data. In: Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Issue 7, pp. 154-–165 (2009)
3. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning Structured Embeddings of Knowledge Bases. In: Proceedings of the 25th Conference on Artificial Intelligence (AAAI-11), San Francisco, USA (2011)
4. Cabrio, E., Aprosio, A. P., Cojan, J., Magnini, B., Gandon, F., Lavelli, A.: QAKiS @ QALD-2. In: Proceedings of the ESWC 2012 workshop Interacting with Linked Data, pp. 87–95. Heraklion, Greece (2012)

5. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1606-1611 (2007)
6. Gerber, D., Ngonga Ngomo, A.-C.: Extracting Multilingual Natural Language Patterns for RDF Predicates. In: Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management, Galway (2012)
7. Hirst, G., St-Onge, D.: Lexical chains as representations of context for the detection and correction of malapropisms. In: Fellbaum, C., Miller G. (eds.) WordNet: An Electronic Lexical Database, pp. 305–332 (1998)
8. Hoffart, J., Berberich, K., Weikum, G.: YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence, vol. 194, pp. 28–61 (2013)
9. Jena ARQ query engine for the SPARQL RDF query language, http://jena.apache.org/documentation/query/index.html
10. Jena TDB RDF store, http://jena.apache.org/documentation/tdb/
11. Klein, D., Manning, C.D.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423–430 (2003)
12. Lopez, V., Unger, C., Cimiano, P., Motta, E.: Evaluating Question Answering over Linked Data. In: Journal of Web Semantics. To appear, Elsevier (2013)
13. MusicBrainz, The Open Music Encyclopedia, http://musicbrainz.org/
14. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: Proceedings of HLT-NAACL 2003, pp. 252–259 (2003)
15. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: Proceedings of the 21st International Conference on World Wide Web (WWW '12), ACM, New York, USA, pp. 639-648 (2012)
16. WS4J (WordNet Similarity for Java), https://code.google.com/p/ws4j/