Evaluation of a layered approach to question answering over linked data

Sebastian Walter[1], Christina Unger[1], Philipp Cimiano[1], and Daniel Bär[2]*

[1] CITEC, Bielefeld University, Germany
http://www.sc.cit-ec.uni-bielefeld.de/
[2] Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de/

Abstract. We present a question answering system architecture which processes natural language questions in a pipeline consisting of five steps: i) question parsing and query template generation, ii) lookup in an inverted index, iii) string similarity computation, iv) lookup in a lexical database in order to find synonyms, and v) semantic similarity computation. These steps are ordered with respect to their computational effort, following the idea of layered processing: questions are passed on along the pipeline only if they cannot be answered on the basis of earlier processing steps, thereby invoking computationally expensive operations only for complex queries that require them. In this paper we present an evaluation of the system on the dataset provided by the 2nd Open Challenge on Question Answering over Linked Data (QALD-2). The main, novel contribution is a systematic empirical investigation of the impact of the single processing components on the overall performance of question answering over linked data.

Keywords: question answering, linked data, layered approach, experimental evaluation

1 Introduction

Question answering over linked data has recently emerged as an important paradigm allowing non-expert users to access the steadily growing amount of data available as linked data (see [9] for a recent overview). One of the main challenges in question answering over linked data is mapping natural language questions into appropriate SPARQL queries or graph patterns that yield an appropriate and correct answer when evaluated. A crucial subtask to this end is to map words in the query to appropriate URIs representing their meaning. For example, when interpreting the question When was Abraham Lincoln born? with respect to the DBpedia dataset, the name Abraham Lincoln needs to be mapped

* Part of this work has been supported by the Klaus Tschira Foundation under project No. 00.133.2008.

to the resource <http://dbpedia.org/resource/Abraham_Lincoln>, and born needs to be mapped to <http://dbpedia.org/ontology/birthplace>.

In this paper, we present a layered approach to question answering over linked data. The main intuition underlying this layered approach is the idea that a question answering system should be sensitive to the complexity of the question, in the sense that it applies certain processing steps only if the question cannot be answered using simpler mechanisms.

To give an idea of the various levels of difficulty, consider the following three questions taken from the DBpedia training questions of the 2nd Open Challenge on Question Answering over Linked Data (QALD-2, see Section 3.1 below):[3]

1. (a) What is the currency of the Czech Republic?
   (b)
```
SELECT DISTINCT ?uri WHERE {
   res:Czech_Republic dbo:currency ?uri .
}
```
2. (a) Who was the wife of U.S. president Lincoln?
   (b)
```
SELECT DISTINCT ?uri WHERE {
   res:Abraham_Lincoln dbo:spouse ?uri.
}
```
3. (a) Was Natalie Portman born in the United States?
   (b)
```
ASK WHERE {
   res:Natalie_Portman dbo:birthPlace ?city .
   ?city dbo:country res:United_States .
}
```

Question 1a exemplifies the simplest case: All natural language expressions can be mapped to DBpedia resources in the target SPARQL query 1b by simply matching the expressions (currency and Czech Republic) with the resources' labels, in this case `currency` and `Czech Republic`. Furthermore, the resources are directly related, so that the SPARQL query consists of one triple relating the entity `Czech Republic` with its currency. This is also the case for query 2b: The entity `Abraham Lincoln` is directly connected to his wife. However, matching the expressions used in the question 2a with DBpedia concepts (U.S. president Lincoln with `Abraham Lincoln`, and wife with `spouse`) is not straightforward but requires searching for synonyms and lexical variants. Similarly in example 3, where the natural language term born needs to be matched with the ontology label `birth place`. Moreover, the property `birth place` does not directly connect the occurring entities, Natalie Portman and the United States; instead they are connected via an intermediate node that is not expressed in the natural language question, the SPARQL query thus has a more complex structure.

As main contribution, we present the results of a systematic evaluation of the contribution of different state-of-the-art processing components on the overall system. For this purpose, we use the benchmarking datasets provided by the QALD-2 challenge. A systematic evaluation of the impact of various components

[3] The following prefixes are used:
```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
```

on the task has so far not been provided. This layered approach also allows us to assess the complexity of the questions in the dataset in terms of which processing is required to actually find an answer. We report the results in Section 3 below. Furthermore, we compare our system to the question answering systems that participated in the QALD-2 challenge as well as to Wolfram Alpha[4].

The paper is structured as follows: In Section 2 we present the architecture of our system in detail. In Section 3 we report on experiments on the QALD-2 dataset, presenting our results in terms of standard precision, recall and F-measure figures for each processing layer, thus being able to quantify the impact in terms of effectiveness of each layer. We also report the average times the approach requires to answer questions depending on the processing depth. This allows for a discussion of the trade-off between effectiveness and efficiency. Finally, we compare our approach to related work in Section 4, before concluding in Section 5.

2 Layered approach

The system we propose, BELA, takes a natural language question as input and produces a SPARQL query as well as corresponding answers as output. It is layered in the sense that it builds on a pipeline along which hypotheses for the meaning of a natural language question are iteratively added and refined by factoring in more and more expensive processing mechanisms. At each layer, the best hypotheses is determined. If the confidence of the system in the hypothesis is high enough and the constructed query actually produces answers, the processing stops and the answers are returned.

BELA processes an incoming natural language question along the following layers:

1 Parsing and template generation
2 Inverted index lookup
3 String similarity computation
4 Lexical expansion
5 Semantic similarity computation

We will describe each of them in more detail in the following sections.

2.1 Parsing and template generation

Each input question is parsed on the basis of its part-of-speech tags, employing a parser based on Lexical Tree Adjoining Grammars (LTAG), in order to produce several query templates for the question. The parser has been described in more detail in [13], so we limit our description to the output of the parser. Parsing a natural language question produces a set of SPARQL query templates corresponding to proto-interpretations of this question, which mirror the semantic structure of the questions and only leave open slots where appropriate URIs need to be inserted. An example is given in 4.

[4] `http://www.wolframalpha.com`

4. (a) What is the currency of the Czech Republic?
   (b) `SELECT ?y WHERE {`
   `?y -- ?p -- ?x`
   `}`
   Slots:
   – ⟨`?p`, unknown, currency⟩
   – ⟨`?x`, resource, Czech Republic⟩

For the question in 4a the template in 4b is constructed. It specifies the overall structure of the query, but leaves open slots for a resource expressed as Czech Republic, which is related to `?y` by means of some property denoted by the noun currency. The dashes indicate that it is left open whether `?y` is subject or object of the property, i.e. whether the triple is `?y ?p ?x .` or `?x ?p ?y .`

2.2 Index lookup

Consider the example 4 above. The first step of processing this template consists in a simple lookup of all slot terms in an inverted index. For indexation, we extract all concepts from DBpedia 3.7 subsumed by the `ontology` and `property` namespace together with their `rdfs:label`, from which we build an inverted index that maps each label to a set of URIs. Additionally, we include Wikipedia re-directs, such that a range of labels, e.g. IBM, I.B.M., International Business Machine and IBM Corporation map to the same URI, in this case `<http://dbpedia.org/resource/IBM>`. The resulting index contains more than 8 million entries: 8,011,004 mappings of labels to resources, 785 mappings of labels to classes, and 92,910 mappings of labels to properties (3,362 from the `ontology` and 89,548 from the `property` namespace).

For the example in 4, the slot terms currency and Czech Republic are found in the index, therefore the following two hypotheses about possible instantiattion of the query slots with URIs are built:

5. – Slot: ⟨`?p`, property, currency⟩
   – URI: `<http://dbpedia.org/ontology/currency>`
   – Rank: 1.0
6. – Slot: ⟨`?x`, resource, Czech Republic⟩
   – URI: `<http://dbpedia.org/resource/Czech_Republic>`
   – Rank: 1.0

The rank is a confidence value between 0 and 1. Here the rank is set to 1, as we take a single direct match in the index to be a sure indicator for a successful mapping. If several mappings are found, a hypothesis for each of them is created (leaving disambiguation to the success or failure of these hypotheses). Also note that the previously unknown type of `?p` can now be specified as property, the type of the found URI.

Using the above hypotheses to instantiate the SPARQL template yields the following two alternative interpretations of the question corresponding to the interpretations of `?y` is subject or object, respectively:

7. (a)
```
SELECT ?y WHERE {
   ?y <http://dbpedia.org/ontology/currency>
   <http://dbpedia.org/resource/Czech_Republic> .
}
```
   (b)
```
SELECT ?y WHERE {
   <http://dbpedia.org/resource/Czech_Republic>
   <http://dbpedia.org/ontology/currency> ?y .
}
```

All generated queries are then sent to the SPARQL endpoint and the highest ranked query that actually returns an answers is selected as final output.[5] In our example case, the query in 7b does return an answers and thus seems to represent a valid interpretation of the natural language question.

In case none of the queries returns an answer, BELA proceeds with the next step.

2.3 String similarity

In case that the basic mechanism of index lookup fails to find appropriate URIs for all slots to produce a completely instantiated SPARQL query, we use identified resources as starting point and retrieve all their properties. As an example, consider the following question and its corresponding template:

8. (a) How many employees does IBM have?
   (b)
```
SELECT COUNT(?y) WHERE {
   ?x -- ?p -- ?y
}
```
   Slots:
   - ⟨?p, property, employees⟩
   - ⟨?x, resource, IBM⟩

An index lookup retrieves `<http://dbpedia.org/resource/IBM>`, which now serves as starting point for finding possible instantiations for the property slot expressed by employees. To this end, we query the dataset for labels of all properties that connect the resource `<http://dbpedia.org/resource/IBM>` to other resources or to literals. For the above example, this yields a list of about 100 properties, including for example `products`, `industry`, `foundation place`, `company type`, `number of employees` and `num employees`.

Next, all retrieved property labels are compared to the slot term, in our example employees, by means of the normalized Levenshtein distance *NLD* between two words w_1 and w_2, calculated as follows:

$$NLD(w_1, w_2) = 1 - \frac{\text{number of letter changes between } w_1 \text{ and } w_2}{\mathbf{max}(length(w_1), length(w_2))}$$

[5] In the case of ASK queries, however, we cannot dismiss queries on the basis of an empty result set as they always return a boolean as answer. Since one concept found in the index is as good as any other concept found in the index, the decision to return a specific query as final result is postponed until the subsequent steps, when query ranks start to vary. Then the highest ranked query above a certain threshold (set to 0.9 in our case) is returned.

All properties that have a label with a Levenshtein distance above a certain threshold, in our case established as 0.95, is added as a new hypothesis with the *NLD* value as its rank. In our case the best matching property is `num employees` with a Levenshtein score of 0.73. Although this is below the threshold, the property label bears strong similarity with the slot term employees, we therefore want to permit it as a hypothesis. To this end, we apply an additional heuristic that assigns rank 1 to a property if its label contains the slot term as a substring.[6] Therefore both the property `<http://dbpedia.org/ontology/numberOfEmployees>` as well as the property `<http://dbpedia.org/property/numEmployees>` is added as hypotheses with rank 1.

Finally, this processing step yields the following query for the question How many employees does IBM have, which retrieves the correct answer:

```
9. SELECT ?y WHERE {
      <http://dbpedia.org/resource/IBM>
      <http://dbpedia.org/ontology/numberOfEmployees> ?y .
   }
```

2.4 Lookup in lexical database

Now consider the question Who is the mayor of Berlin. Both layers described above—index lookup and string similarity—do not find an answer to this question as the right interpretation involves the property `<http://dbpedia.org/ontology/leader>` rather than a property with label `mayor`. Therefore, in a third processing step, we use a lexico-semantic resource, in this case WordNet, to retrieve synonyms for slot terms. The slot term mayor, for example, leads to a list containing civil authority, politician, ex-mayor and city manager, among others. While tuning BELA on the QALD-2 training question set for DBpedia, we found that the overall results improve if this list is further expanded with the synonyms, hypernyms and hyponyms of all list elements; in our example this adds authority, leader, governor and judge, among others. These synonyms are matched to all properties retrieved for the resources explicitly mentioned in the question (`<http://dbpedia.org/resource/Berlin>`in the example), and in case of a match, an appropriate hypothesis is generated. In the case of the above question, this leads to the following correct SPARQL query:

```
10. SELECT ?y WHERE {
       <http://dbpedia.org/resource/Berlin>
       <http://dbpedia.org/ontology/leader> ?y .
    }
```

2.5 Semantic similarity

In case the string similarity and lexical expension steps do not find sufficiently high ranked hypotheses, BELA tries to find suitable hypotheses by means of *Explicit Semantic Analysis* (ESA).[7]

ESA is a method introduced by Gabrilovich and Markovitch [5] in order to represent and compare texts of any length in a high-dimensional vector space. The vector space is

[6] The rank is set to 1 in order to push these hypotheses above the Levenshtein threshold of 0.95 and to make them fare better than purely string similar hypotheses.

[7] The implementation we used is available at: `http://code.google.com/p/dkpro-similarity-asl/`.

constructed based on a given document collection D, where the documents are assumed to describe natural concepts such as *cat* or *dog* (a so-called concept hypothesis). In the construction phase, a term-document matrix is built with a *tf.idf* weighting scheme [12] of terms w.r.t. the documents $d \in D$. A semantic interpreter then allows to map any given natural language text t onto concept vectors: Each word $w \in t$ is represented by the concept vector $\boldsymbol{c}(w)$ of the corresponding row in the term-document matrix, where each vector element denotes the strength of association with a particular document $d \in D$. For $|t| > 1$ (i.e., texts rather than single words) the vector $\boldsymbol{c}(t)$ constitutes the sum of the individual word vectors $\boldsymbol{c}(w)$ for all $w \in t$. Finally the two concepts vectors are compared using cosine similarity, thus yielding a semantic similarity score for the compared texts. While in the original work of Gabrilovich and Markovitch (2007) Wikipedia was used as background knowledge source, recent work has shown that also Wiktionary[8] and WordNet [4] can be used as background document collections. Initial experiments showed that using Wikipedia as background document collection produces the best results on tour task. In the following, all experiments involving ESA are thus carried out using Wikipedia as background knowledge base.[9]

Applying ESA to the question answering task allows us to relate, e.g., the expression `painted` and the ontology label `artist`, which fail to be connected by both string similarity and WordNet expanison.

3 Experiments

3.1 Evaluation set-up

BELA has been evaluated on the DBpedia training and test question sets provided by the 2nd Open Challenge on Question Answering over Linked Data[10] (QALD-2). A more detailed description of these datasets and the procedure for constructing it can be found in [8]. Both datasets comprise 100 natural language questions annotated with SPARQL queries and answers. From these questions we removed all out-of-scope questions (questions that cannot be answered within the dataset) as well as questions relying on namespaces not yet part of our index, namely YAGO and FOAF. This filtering led to remaining 75 training and 72 test questions. In order to ensure a fair evaluation, we used only the training set for developing and fine-tuning the system, e.g. for determining the Levenshtein and ESA thresholds, and used the test set for the purpose of evaluation only. By manual tuning on the training dataset, the threshold for the normalized Levenshtein distance was set to 0.95, while the threshold for ESA was set to 0.4.

For evaluation we used the tool provided by the QALD-2 challenge. For each question q, precision, recall and F-measure are computed as follows:

$$Recall(q) = \frac{\text{number of correct system answers for } q}{\text{number of gold standard answers for } q}$$

$$Precision(q) = \frac{\text{number of correct system answers for } q}{\text{number of system answers for } q}$$

[8] `http://www.wiktionary.org`

[9] Results for all tested dictionaries can be found at `http://www.sc.cit-ec.uni-bielefeld.de/bela`.

[10] `http://www.sc.cit-ec.uni-bielefeld.de/qald-2`

Module	Answered	Coverage	Correct	R	P	F	F'
DBpedia Train							
Index lookup	15	0.2	7	0.67	0.61	0.64	0.30
+ String similarity	29	0.38	16	0.77	0.73	0.75	0.50
+ Lexical expansion	37	0.49	20	0.74	0.69	0.71	0.57
+ Semantic similarity	39	0.52	22	0.75	0.71	0.73	0.60
DBpedia Test							
Index lookup	11	0.15	9	0.909	0.84	0.87	0.25
+ String similarity	20	0.27	13	0.85	0.74	0.79	0.40
+ Lexical expansion	29	0.4	16	0.71	0.63	0.67	0.50
+ Semantic similarity	31	0.43	17	0.73	0.62	0.67	0.52

Table 1. Results over the 75 DBpedia train and 72 DBpedia test questions

$$\textit{F-Measure}(q) = \frac{2 * \textit{Precision}(q) \times \textit{Recall}(q)}{\textit{Precision}(q) + \textit{Recall}(q)}$$

On the basis of these, overall precision and recall values P and R, as well as an overall F-measure value F are computed as the average mean of the precision, recall and F-measure values for all questions. Additionally, we compute coverage as the percentage of questions for which an answer was provided: $\textit{Coverage} = \frac{\text{number of queries with answer}}{|Q|}$. In order to also take into account the balance between F-measure and coverage, we introduce an F-measure F' as the harmonic mean of the coverage and the overall F-Measure $F' = \frac{2 \times \textit{Coverage} \times F}{\textit{Coverage} + F}$.

3.2 Results

Table 1 shows the results on the DBpedia training and test sets. It lists the number of answered queries, the coverage, the number of questions that were answered perfectly as well as the average precision, recall and F-measures F and F'.[11] The behavior of the system is as expected in the sense that the overall performance (F') increases with each processing step in the pipeline, where string similarity computation clearly has the most impact on the results, increasing performance by 20% on train and 15% on test. The use of a lexical database (in our case WordNet) increases the results by 7% on train and 10% on test, followed by the semantic similarity component, which increases results by 3% on train and 2% on test. Thus all components provide an added value to the overall pipeline. Table 2 lists the number of questions that can be answered at a certain processing step in the pipeline but could not be answered earlier.

3.3 Comparison with state-of-the-art systems

Table 3 compares the results of our system BELA (traversing the full pipeline) with the results of the systems that participated in the QALD-2 challenge and with Wolfram Alpha.[12] In addition to the number of correctly answered questions, we list the number

[11] A more detailed listing of the results for each question can be found at `http://www.sc.cit-ec.uni-bielefeld.de/bela`.

[12] In order to allow for a comparison with Wolfram Alpha, we submitted the test questions to the Wolfram Alpha portal and extracted and verified the results manually.

Module	Train	%	Test	%
String similarity	24	61	20	64
Lexical expansion	10	25	11	35
Semantic similarity	2	5	2	6

Table 2. How many questions require the pipeline up to which module to be answered?

of questions for which a partially correct answer was provided, i.e. questions with an F-measure strictly between 0 and 1.

The comparison shows that BELA ranges, from the point of view of overall performance, in the middle field, outperforming Wolfram Alpha in particular. The main difference between our system and the two systems that outperform it—SemSeK and MHE—is that the latter achieve a much higher coverage at the price of a much lower precision.

System	Answered	Coverage	Correct	Partially	R	P	F	F'
SemSeK	80	0.8	32	7	0.44	0.48	0.46	**0.58**
MHE	**97**	**0.97**	30	12	0.36	0.4	0.38	0.54
BELA	31	0.31	17	5	**0.73**	**0.62**	**0.67**	0.42
WolframAlpha	51	0.51	15	2	0.32	0.3	0.309	0.38
QAKiS	35	0.35	11	4	0.39	0.37	0.38	0.36
Alexandria	25	0.25	5	10	0.43	0.46	0.45	0.32

Table 3. Results compared with results from the participants of the QALD-2 challenge (with coverage calculated over all 100 questions)

3.4 Performance

The following table shows the average time for answering a question (in seconds, calculated over the train and test dataset):[13]

Index lookup	+ String similarity	+ Lexical expansion	+ Semantic similarity
4.5	5.2	5.4	16.5

The average time for answering a question, to no or little surprise, increases when increasing the number of modules used by the system. However, the average cost of the index lookup, string similarity and lexical expansion steps is very similar; a significant increase in fact arises only when adding semantic similarity to the computation, raising the average time per second by around 11 seconds, while providing only a 2% performance increase.

The parsing and template generation step takes an average of 1.7 seconds per question. In future work, we will optimize the index lookup and pre-caching mechanism, now taking up an average of two seconds per questions. Saving the pre-cached informations after an experiment, the average time in the next experiment drops to around 4.6 second per question for the second and third step of the pipeline.

[13] Performed on a machine with a Intel® Core™ i3-2310M CPU @ 2.10GHz.

3.5 Manual, query-driven extension of lexical coverage

Although similarity and relatedness measures can bridge the gap between natural language terms and ontology labels to a certain extent, they fail when the gap is too big. For example, all modules included in BELA failed to relate `created` and `author`, or `die` and `deathCause`. Now, mappings that are notoriously difficult to find for a machine could be easy to create by someone with basic domain knowledge. Considering, for example, a question answering system that logs the questions it fails to answer, a maintainer could manually specify index mappings for natural language expressions that are often used.

In order to show how little manual effort is required to increase precision and recall, we additionally report on a run of the full pipeline of the system enriched with an additional, manually created index that contains 14 mappings from natural language terms to URIs which BELA failed to identify, for example `high` → `<http://dbpedia.org/ontology/elevation>`.[14] Given such a manual index with 14 entries, the results increase, as shown in Table 4. Note that even the results on the test question set slightly increase, although when building the manual index only training questions were taken into account. Thus a relatively small manual effort can help bridging the gap between natural language expressions and ontology labels in case similarity and relatedness measures fail.

	Answered	Coverage	Correct	Partially	R	P	F	F'
Train (without)	39	0.52	22	11	0.75	0.71	0.73	0.60
Train (with)	40	0.53	26	10	0.80	0.81	0.80	0.63
Test (without)	31	0.43	17	6	0.73	0.62	0.67	0.52
Test (with)	32	0.44	18	6	0.74	0.639	0.688	0.53

Table 4. Results of full pipeline with manually extended index

4 Discussion and related work

We can identify two major challenges when constructing SPARQL queries for natural language questions:

- Briding the *lexical gap*, i.e. the gap between natural language expressions and ontology labels (e.g. `mayor` and `leader`, `written` and `author`)
- Bridging the *structural gap*, i.e. the gap between the semantic structure of the natural language question and the structure of the data

The lexical gap is quite well covered by the tools exploited in our pipeline, i.e. string similarity as well as lexico-semantic resources and semantic relatedness measures, all of which are quite standard in current Semantic Web question answering systems. An additional, recently emerging tool for bridging the lexical gap are repositories of natural language representations of Semantic Web predicates, acquired from a structured data repository together with a text corpus. Examples are the BOA pattern library [6] (used,

[14] The complete list can be found at `http://www.sc.cit-ec.uni-bielefeld.de/bela`.

e.g., in TBSL [13]) and the WikiFramework repository [10] (used, e.g. in QUAKiS [3]). Both go beyond semantic similarity measures in also involving co-occurence patterns.

The structural gap, on the other hand, is less often addressed. Most systems map natural language questions to triple-based representations and simply fail if this representation does not match the actual data. A simple example is the query Give me all cities in Germany. Our system starts looking for resources of class city that are directly related to the entity Germany; in the actual data, however, some cities are only indirectly connected to their country, e.g. through their federal state. Such a case requires a search for indirect relationships in case direct ones cannot be found. *PowerAqua* [7], an open-domain question answering system for the Semantic Web, does exactly this. After mapping natural language questions to a triple-based representation and discovering relevant ontologies, PowerAqua first tries to find entity mappings, exploiting different word sense disambiguation techniques. Then it searches for direct relationships between the candidate entities, using WordNet expansion and also different filtering heuristics to limit the search space. If no direct relationships are found, indirect relationships are explored.

A slightly more difficult example is the question When did Germany join the EU. Our template generation process assumes a representation with two entities, Germany and the EU, and a relation join connecting them; PowerAqua[15] assumes a triple representation of form ⟨`date,join,Germany`⟩,⟨`Germany,?,EU`⟩. The actual DBpedia data, however, relates Germany to a date literal via the property `accessioneudate`, thus both representations fail to match it. Such cases therefore require more sophisticated techniques for inferring or learning the target triple structure from the data or an underlying ontology. In order to bridge the structural gap, a system architecture like ours would therefore require further iterations: Once the whole pipeline is traversed without having constructed a successful query, the template structure needs to be adapted or extended, triggering a new pipeline cycle.

We conjecture that a proper approach to bridging the structural gap is necessary to further increase the coverage and performance of question answering systems significantly, and that without such an approach, comprehensive question answering over linked data will fail, just like without a proper approach to bridging the lexical gap.

Another major challenge for question answering over linked data is the processing of questions with respect to not only one but several datasets (ultimately the whole linked data cloud), which includes the search for relevant ontologies as well as the integration of query parts constructed from different sources. This challenge has so far only been taken up by PowerAqua. Also, evaluating and comparing question answering systems in such an open-domain scenario is inherently difficult.

5 Conclusion

We have presented a layered architecture for question answering over linked data that relies on an ordered processing pipeline consisting of the following steps: an inverted index lookup, the computation of string similarities, a lookup in a lexical database such as WordNet and a semantic similarity computation step based on Explicit Semantic Analysis. We have systematically evaluated the contribution of each of these component on the benchmarking dataset provided by the 2nd Open Challenge on Question

[15] Accessed through the online demo at `http://poweraqua.open.ac.uk:8080/poweraqualinked/jsp/`.

Answering over Linked Data (QALD-2), showing that each of these processing components has an important impact on the task, increasing coverage and F-measure while obviously increasing the overall processing time. We have also shown that our approach can compete with other state-of-the-art systems, e.g. clearly outperforming Wolfram Alpha. Finally, we have shown how an iterative improvement lifecycle that adds additional mappings to the system can substantially improve the performance of the system. Future work will consider adding additional lexical knowledge to the system (e.g. Wiktionary and lexical pattern libraries), and will especially focus on adding iterations that adapt the structure of the query templates, in order to bridge the gap between the semantic structure of the natural language question and the structure of the dataset.

References

1. M. Anderka, B. Stein: The ESA Retrieval Model Revisited. In: Proc. of the 32th Annual International ACM SIGIR Conference (2009) 670–671
2. C. Bizer, T. Heath, T. Berners-Lee: Linked Data – The Story So Far. Int. Journal on Semantic Web and Information Systems **5** (2009)
3. E. Cabrio, A. Palmero Aprosio, J. Cojan, B. Magnini, F. Gandon, A. Lavelli: QAKiS @ QALD-2. In: Proc. of ILD 2012, `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/2/proceedings_ILD2012.pdf`
4. C. Fellbaum: WordNet: An Electronic Lexical Database. MIT Press (1998)
5. E. Gabrilovich, S. Markovitch: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: Proc. of the 20th International Joint Conference on Artificial Intelligence (2007) 1606–1611 ibitemqald V. Lopez, C. Unger, P. Cimiano, E. Motta: Evaluating Question Answering over Linked Data. Journal of Web Semantics, under review.
6. D. Gerber, A.-C. Ngonga Ngomo: Bootstrapping the Linked Data Web. In: Proc. of WekEx at ISWC 2011.
7. V. Lopez, M. Fernández, E. Motta, N. Stieler: PowerAqua: supporting users in querying and exploring the Semantic Web content. Semantic Web journal, to appear. Available from `http://www.semantic-web-journal.net/`
8. V. Lopez, C. Unger, P. Cimiano, E. Motta: Evaluating Question Answering over Linked Data. Journal of Web Semantics (under review)
9. V. Lopez, V. Uren, M. Sabou, E. Motta: Is Question Answering fit for the Semantic Web? A Survey. Semantic Web Journal **2** (2011) 125–155
10. R. Mahendra, L. Wanzare, R. Bernardi, A. Lavelli, B. Magnini: Acquiring Relational Patterns from Wikipedia: A Case Study. In: Proc. of the 5th Language and Technology Conference (2011)
11. G.A. Miller: WordNet: A Lexical Database for English. In: *Communications of the ACM* **38** (1995) 39–41
12. G. Salton, M.J. McGill: Introduction to Modern Information Retrieval. McGraw-Hill (1983)
13. C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, P. Cimiano: Template-Based Question Answering over RDF data. In: Proc. of WWW 2012
14. T. Zesch, C. Müller, I. Gurevych: Using Wiktionary for Computing Semantic Relatedness. In: Proc. of the 23rd AAAI Conference on Artificial Intelligence (2008) 861–867