Linguistic Modeling of Linked Open Data for Question Answering

Matthias Wendt, Martin Gerlach, and Holger Düwiger

Neofonie GmbH, Robert-Koch-Platz 4, 10115 Berlin, Germany
{wendt,gerlach,duewiger}@neofonie.de,
WWW home page: http://www.neofonie.de/Forschung

Abstract. With the evolution of linked open data sources, question answering regains importance as a way to make data accessible and explorable to the public. The triple structure of RDF-data at the same time seems to predetermine question answering for being devised in its native subject-verb-object form. The devices of natural language, however, often exceed this triple-centered model. But RDF does not preclude this point of view. Rather, it depends on the modeling. As part of a government funded research project named Alexandria, we implemented an approach to question answering that enables the user to ask questions in ways that may involve more than binary relations.

Introduction

In recent years, the Semantic Web has evolved from a mere idea into a growing environment of Linked Open Data (LOD)[1] sources and applications. This is due in particular to two current trends: The first is automatic data harvesting from unstructured or semi-structured knowledge that is freely available on the internet, most notably the DBpedia project [1]. The second notable trend is the evolution of linked data sources with possibilities of collaborative editing such as Freebase[2]. The growth of LOD gives rise to a growing demand for means of semantic data exploration. Question Answering (QA), being the natural device of querying things and aqcuiring knowledge, is a straightforward way for end users to access semantic data.

RDF[3] and other languages for triple-centered models, which are often used to model and describe linked data, seem to predetermine a specific way of thinking - and of asking questions. Many RDF sources offer information in the form "X birth-place Y" and "X birth-date Z", etc. Of course, in natural language, we are used to formulate complexer queries. It is natural to make statements like "X was born in Y on Z". While this does not matter as long as singular events like birth or death are involved, things become more complicated as soon as events are involved that can occur more than once. For example, the question "Who was married to Angelina Jolie in 2006?" can only be answered if the temporal (and potentially limited) nature of a relation like marriage is taken into account.

[1] See http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[2] http://www.freebase.com/

[3] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

In this paper we present the QA-driven ontology design behind Alexandria[4], a platform for exploring data of public interest comprising a system for answering questions in German. The domain consists of persons, organizations, locations as well as works such as books, music albums, films and paintings. Moreover, the Alexandria ontology is designed for holding information on events relating the various resources, including temporal information and relations involving more than two participants – so called N-ary Relations. Also, we describe the mapping algorithm used in our question answering system and how it benefits from the ontology design.

The ontology is built from and continuously being updated with data primarily from Freebase, and few parts from DBpedia, news feeds, and user generated content.

Related Work

Open domain question answering is of current research interest. There are several approaches to the subject based on linguistic analysis of natural language questions for generating queries against linked data.

FREyA [5] and PowerAqua [9,10] are both question answering systems that are to a certain degree independent of the underlying ontology schema. Both systems work on existing Linked Open Data *as is* and can be configured to use multiple ontologies. They rely on rather shallow approaches to query mapping, in favor of portability and schema-independence. However, this also limits them to the data structures and languages used by the schemas (e.g., DBpedia does not support N-ary relations).

There are also systems based on deeper, compositional mapping approaches. For example, ORAKEL [3,4] translates syntax tree constructed by lexicalized tree adjoining grammars (LTAGs) to a representation in first order logic which can be converted to F-Logic [8] or SPARQL[5] queries, depending on the target knowledge base. ORAKEL also principally supports N-ary relations. Though the system is in principle very similar to the one presented in this paper, it is not proven to scale up to a large data set.

In contrast to other projects that use Linked Open Data for question answering, our approach is an attempt to combine the advantages of availability of huge LOD sources and of tailoring the T-Box to the use case of QA. While the latter facilitates the fully automated mapping of natural language questions to SPARQL queries, we trade off the possibility to use existing labels for T-Box entities, which, combined with existing lexical resources such as WordNet[6], GermaNet[7], etc., boost lexical coverage.

Another difference to the above-mentioned projects is that the focus of Alexandria is on answering questions in German, not English.

Design of the Alexandria Ontology

The design of the Alexandria Ontology was basically driven by practical demands of the application as well as linguistic considerations. According to [7], our approach can be seen as a unification of the "Type 4" and the "Type 3" approaches to ontology creation. The knowledge base has to meet the following requirements:

[4] `http://alexandria.neofonie.de/`
[5] `http://www.w3.org/TR/rdf-sparql-query/`
[6] `http://wordnet.princeton.edu/`
[7] `http://www.sfs.uni-tuebingen.de/lsd/`

Linguistic Suitability The data model needs to be suitable for natural language question answering, i.e. mapping natural language parse tree structures onto our data must be possible.

LOD Compatibility Compatibility with existing LOD sources like Freebase and DBPedia needs to be maintained in order to facilitate mass data import for practical use.

Scalability Large amounts of data need to be stored, maintained and updated while keeping the time for answering a question at minimum.

One of the major aspects relating to *linguistic suitability* in the Alexandria use case is that its target domain goes beyond what we refer to in the following as *attributive data*, i.e. data about things that are commonly known as named entities like persons, organizations, places, etc. In addition, the domain was designed to contain what we call *eventive data*, i.e. (historic) events and relations to participants within them.

As mentioned above, there are certain relations, such as *birth*, where this distinction is not important, because n-ary relations consisting of unique binary parts (like place and date of birth) can be covered by joining on a participant (the person) as proposed in [4]. The distinction between eventive and attributive data becomes important when relations are involved, which (may) occur repetitively and/or include a time span. Questions like "Who was married to Angelina Jolie in 2001?" and "Which subject did Angela Merkel major in at the German Academy of Sciences?" can no longer be generally answered by joining binary facts.

It is possible to model such eventive n-ary facts as proposed in Pattern 1, use case 3 of the W3C Working Group Note on N-ary Relations on the Semantic Web[8]. This approach is also close to the semantic model advocated in Neo-Davidsonian theories [12], where participants in an event are connected to the event using roles.

As for the aspect of *LOD compatibility*, it is our aim to access existing large-scale sources to populate our knowledge base. DBpedia was the first LOD source to retrieve and constantly update its data repository by crawling Wikipedia[9]. Apart from its possibilities for end-users to add and update information, the majority of data contained in Freebase is obtained from Wikipedia as well. Therefore, using one (or both) of these sources is an obvious starting point for harvesting information on a broad range of popular entities, as it is required by Alexandria.

However, though DBpedia contains much valuable attributive data for entities of our interest, it does not offer eventive information as stated above. Also, DBpedia's T-Box does not provide a model for adding such n-ary facts, either.

As opposed to DBpedia, which relies on the RDF standard, Freebase implements a proprietary format. Whereas in RDF, all information is abstractly represented by triples, Freebase abstractly represents information as links between topics. The Freebase data model incorporates n-ary relations by means of Compound Value Types[10], also called "Mediators". A mediator links multiple topics and literals to express a single fact.

So Freebase's data model suits our requirements, but we need to use RDF to be able to use Virtuoso Open Source Edition[11] which has proven to *scale well* for both loading and querying the amounts of data we expected.

[8] `http://www.w3.org/TR/swbp-n-aryRelations/`
[9] `http://www.wikipedia.org/`
[10] `http://wiki.freebase.com/wiki/Compound\_Value\_Type`
[11] `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/`

Using the Freebase query API to pull a set of topics and links, an RDF based knowledge base can be built according to the Neo-Davidsonian model. The API also supports querying link updates for continously updating the knowledge base.

There are straightforward mappings of Freebase topic and mediator types onto OWL[12] classes, and of Freebase link types onto OWL properties. For example, a marriage relation is imported from Freebase as follows:

```
nary:m_02t82g4 rdf:type     dom:Marriage ;
               dom:spouse   res:Angelina_Jolie ;
               dom:spouse   res:Brad_Pitt ;
               alx:hasStart "2005"^^xsd:gYear .
```

The subject URI is generated from the Freebase mediator ID. The resource URIs are generated from Freebase topic names with some extra processing and stored permanently for each Freebase topic ID.

We differentiate the following three layers of our ontology (which correspond to three namespace prefixes alx:, dom:, and res: that appear in the examples):

The Upper Model (alx:) contains the abstract linguistic classes needed for a language-, domain- and task-independent organization of knowledge. The Alexandria upper model is inspired by [6].

The Domain Model (dom:) contains the concrete classes and properties for entities, events and relations of the modeled domain (e.g., Marriage, Study) as subclasses of the upper model classes and properties. Needed to make the domain-specific distinctions which are necessary for the task of question answering.

The A-Box (res:) consists of all "resources", i.e. entity, event and relation instances, known to Alexandria.

The examples of Angela Merkel's education and Angelina Jolie's and Brad Pitt's marriage, which we used above, would be represented as shown in Table 1.

Upper model concept	**Domain concept**	**U.m. role props.**	**Domain props.**	**Participant**
agentive process	study	agent	student	Angela Merkel
effective process	study	affected	subject	Quantum Chemistry
locative relation	study	location	institution	German Academy of Sciences
		located	student	Angela Merkel
attributive rel.	marriage*	carrier	spouse	Brad Pitt
		attribute	spouse	Angelina Jolie
temporal concept	marriage	start	wedding date	2005

* In this example, marriage is modeled as a symmetric relation, expressing a spouse as attribute of the other spouse, i.e. carrier and attribute may be swapped.

Table 1. Upper and domain model

[12] Web Ontology Language, builds on RDF, see `http://www.w3.org/TR/owl2-overview/`

Syntactically, we model our domain concepts as OWL subclasses of one or more upper model concepts and our domain properties as OWL subproperties of one or more upper model properties, where the latter correspond to the Neo-Davidsonian roles mentioned earlier. We can then obtain hints to the upper model concepts and roles of interest by mapping question verbs onto domain concepts and then try to match the roles defined for the upper model classes to the respective given parts of the question.

Putting the Model in Action: Question Answering

As mentioned above, one of the major design goals of our ontology schema was to stay reasonably close to the phenomena and structure of natural language. Achieving this would facilitate the mapping of a natural language question to a SPARQL graph pattern that conveys the information need expressed in the question. The basic idea of the translation algorithm is to understand the problem of mapping of natural language to SPARQL as a graph mapping problem.

From a linguistic viewpoint, the syntactic structure of a sentence may be represented in the form of a dependency tree, as obtained by the application of a dependency parser. A dependency graph is formalized like this:

Given a set L of dependency labels, a *dependency graph* for the sentence $x = w_1 \dots w_n$ is a directed graph $D = (V_D, E_D)$ with:

1. V_D is a set of vertices $\{w_0, w_1, \dots, w_n\}$ and
2. $E_D \subseteq V_D \times L \times V_D$ a set of labeled edges

The vertices are composed of the tokens in the sentence plus an artificial root node w_0. A well-formed dependency graph is a tree rooted at w_0.

Likewise, the structure of a SPARQL Select query basically consists of a graph pattern (in the Where clause) and a projection. Given a set of variable names N_V (`?x`, `?y` ...), the set of concept names N_C, a set of role names N_P, a set of resource names N_R and a set of literals N_L defined by the ontology, we define a SPARQL Select Graph $G = (V_G, E_G, P_G)$ as:

1. $V_G \subseteq N_V \cup N_R \cup N_C \cup N_L$
2. $E_G \subseteq N_V \cup N_R \times N_P \times V_G$
3. the projection $P_G \subseteq N_V$

Formally, we define the translation as a mapping $f(D)$ of a dependency graph D to a SPARQL Select Graph G.

Linguistic Processing

The dependency graph is the result of the application of a **linguistic analysis** to the input sentence. An example of a resulting dependency structure may be found on the left in Figure 1. The analysis consists in tokenization, POS-tagging[13] and dependency parsing. Dependency parsing is conducted using the MaltParser [11], which was trained

[13] For German tokenization and POS-Tagging we use OpenNLP with some pre-trained models. (`http://incubator.apache.org/opennlp/`)

on the German Tiger corpus[14] [2]. The corpus has been slightly adapted by adding a small sub-corpus of German questions and a minor change to the set of role labels used.

To normalize surface form variation and identify morphosyntactic features, lemmatization and morphological analysis is applied to each of the tokens. This is roughly illustrated by the lemmata in square brackets at the verbal nodes (e.g. "verheiratet" has the lemma "verheiraten").

Compositional Semantics

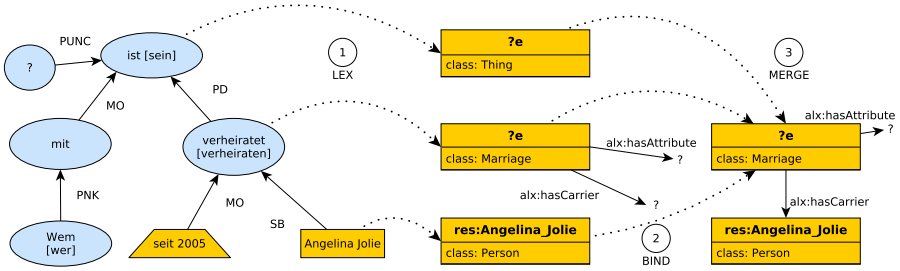

Fig. 1. Dependency parse of the sentence "Mit wem ist Angelina Jolie seit 2005 verheiratet?" and examples of the lexicalization (1) for a subset of its nodes, and the application of the actions BIND (2) and MERGE (3).

The mapping of the dependency graph to the SPARQL query is largely done in two steps: *lexicalization* and *composition*. By *lexicalization* we refer to the process of mapping tokens (lemmata) or multi-word units to corresponding ontological units.

We refer to the identification of resources (identified by resource URIs) of the A-Box as *lexical named entity identification*. For this, we make use of the title (the name of an entity) and the alternative names (consisting of synonyms and different surface forms) that are imported from Freebase into a Lucene[15] index containing the resource URI in Alexandria (e.g. `res:Angelina_Jolie`), and the OWL classes it belongs to. While the user enters a question, matching entities are looked up in the index based on whole words already entered and a disambiguation choice is continuously updated. The user can select from the found entities at any time, whereupon the respective part of the question is updated.

The second noteworthy component in lexical named entity identification is the identification of dates (and time). For these, we have adapted the open source date parser provided by the Yago project[16] to German.

All other linguistic tokens or configurations (linguistic units) corresponding to T-Box concepts are mapped using hand-crafted lexica.

The complete set of mappings for the question shown in Fig. 1 is shown in Table 2.

[14] `http://www.ims.uni-stuttgart.de/projekte/TIGER/`
[15] `http://lucene.apache.org/`
[16] `http://www.mpi-inf.mpg.de/yago-naga/javatools/`

T-Box Class	**T-Box Role**	**A-Box URI**	**Literal**	**T-Box Class**	
Custom Lexica		Lucene	Date, Literal Parser	Custom Lexica	
"wer"	"mit"	"Angelina Jolie"	"2005"	"verheiraten"	"sein"
`dom:Person`	`alx:hasAttribute`	`res:Angelina_Jolie`	`"2005"^^xsd:date`	`dom:Marriage`	`owl:Thing`

Table 2. Types of Lexical Mappings

Our syntax-semantics mapping is largely done by the composition of the lexical semantic entries attached to each dependency node. This lexicalized approach devises the notion of a *semantic description*. A semantic description represents the semantic contribution of a dependency node or (partial) dependency tree and encodes obligatory semantic complements (slots). During the composition, the slots are being filled by semantic descriptions (properties) until the semantic description is satisfied.

By virtue of the lexical mapping each linguistic unit is mapped to a set of semantic descriptions, also called *readings*.

Given a set of variable names N_V, the set of concept names N_C, a set of role names N_P, a set of resource names N_R and a set of literals N_L defined by the ontology, a semantic description S of an ontological entity $n \in N_V \cup N_L \cup N_R$ is defined as a five-tuple $S = (n, c, Sl, Pr, Fl)$ with:

1. $c \in N_C$ the concept URI of the semantic description
2. $Sl = [r_1, r_2, \ldots, r_n]$ a ordered set of slots ($r_i \in N_P$)
3. $Pr = \{(p_1, S_1), \ldots (p_m, S_m)\}$ with S_j a semantic description and $p_m \in N_P$
4. $Fl \subseteq \{proj, asc, desc\}$ a set of flags (with $proj$ indicating that n to be part of the projection of the output graph

For convenience, we define the following access functions for the semantic descriptions $S = (n, c, Sl, Pr, Fl)$:

1. $node(S) = n \Leftrightarrow S = (n, c, Sl, Pr, Fl)$
2. $pred(S) = \{p | (p, o) \in Pr\} \Leftrightarrow S = (n, c, Sl, Pr, Fl)$

A semantic description $S = (n, c, Sl, Pr, Fl)$ is well-formed if the set of bound properties and the slots are disjoint $pred(S) \cap Sl = \emptyset$ and all bound properties are uniquely bound $\forall (p, o) \in Pr \rightarrow \neg\exists (p, o_1) \in Pr \wedge o \neq o_1$.

By definition, there is a strong correlation between a semantic description and a SPARQL Select query. A SPARQL Select query can recursively be built from a semantic description $S = (n, c, Sl, Pr, Fl)$ and an initially empty input graph $G_0 = (V_{G_0} = \emptyset, E_{G_0} = \emptyset, P_{G_0} = \emptyset)$:

toSPARQL(S, G_0): G_m
$G_0 = (V_0 \cup \{n\}, E_0, P_0)$: the output SPARQL graph pattern
$E_0 = E_s \cup \{(n, \mathtt{a}, c)\}$
$P_0 \leftarrow P \cup \{n\} \Leftrightarrow proj \in Fl$ otherwise $P_o = P$

foreach (p_i, o_i) **in** $(p_1, o_1), \ldots, (p_m, o_m) = Pr$
begin
  $E_i \leftarrow E_{i-1} \cup (n, p, node(o))$
  $G_i \leftarrow$ toSPARQL(o_i, G_{i-1})
end
return G_m

To give an example in an informal notation, the linguistic units of the sentence "Mit wem ist Angelina Jolie seit 2005 verheiratet?" are displayed in Table 3. The first row shows the linguistic unit, the ontological unit described corresponds to the variable or resource URI in the second row. The prefix `?!` in a variable designation (e.g. `?!x`) is equivalent to the flag *proj*, denoting that the variable will be part of the projection, i.e. $proj \in Proj$. Note that the verbal nodes "verheiratet" and "sein" are each mapped to a (distinct) variable `?e`, which corresponds to the Neo-Davidsonian event variable e.

Slots and bound properties are displayed in the third column. The slots are designated with the argument being just a `?`, whereas a variable is denoted by the prefix '`?`' and a lower case letter (e.g. `?v`). In the example above the semantic descriptions for "verheiratet" and "mit" contain the slots `alx:hasCarrier` (verb only) and `alx:hasAttribute`.

"Angelina Jolie"	`res:Angelina_Jolie`	`a dom:Person`
"seit 2005"	`?e`	`a alx:TemporalRelation ;` `alx:hasStart 2005 .`
"mit"	`?e`	`a alx:AttributiveRelation ;` `alx:hasAttribute ?`
"wem"	`?!x`	`a dom:Person`
"sein"	`?e`	`a alx:AttributiveRelation`

"verheiratet"	`?e`	`a dom:Marriage ;`
		`alx:hasCarrier ? ;`
		`alx:hasAttribute ?`

Table 3. Semantic descriptions of lexical units.

Putting it all Together

The composition algorithm devises a fixed set of two-place composition operators, called actions. An action defines the mapping of two semantic descriptions related to an edge in the dependency graph to a composed semantic description, corresponding to the semantics of the subgraph of the dependency tree.

The two most important actions are BIND and MERGE. These two basic operations on the semantic descriptions involved in the composition intuitively correspond to (1) the mapping of the syntactic roles to semantic roles (otherwise called semantic role labeling) and (2) the aggregation of two nodes to one in the output graph pattern. Given two semantic descriptions $S_1 = (n_1, c_1, Sl_1, Pr_1, Fl_1)$ and $S_2 = (n_2, c_2, Sl_2, Pr_2, Fl_2)$, the semantic operators are defined like this:

$$BIND(S_1, S_2) = \begin{cases} S(v, c_1, Sl_1 - \{\max(Sl_1)\}, Pr_1 \cup \{(\max(Sl_1), S_2)\}) \\ \qquad \text{if } \text{range}(\max(Sl_1)) \sqcap c_2 \neq \bot \\ NULL \qquad \text{otherwise} \end{cases}$$

$$MERGE(S_1, S_2) = \begin{cases} S(v, lcs(c_1, c_2), Sl_1 \cup Sl_2, Pr_1 \cup Pr_2) \\ \qquad \text{if } c_1 \sqcap c_2 \neq \bot \\ \qquad \wedge pred(S_1) \cap pred(S_2) = \emptyset \\ NULL \qquad \text{otherwise} \end{cases}$$

The function $lcs(c_1, c_2)$ gives the least common subsumer of two concepts, i.e. a concept c such that $c_1 \sqsubseteq c$ and $c_2 \sqsubseteq c$ and for all e such that $c_1 \sqsubseteq e$ and $c_2 \sqsubseteq e$ then $c \sqsubseteq e$. The semantic role labeling implemented by BIND depends on a total order of semantic roles, which has to be configured in the system, e.g.:

`alx:hasAgent` > `alx:hasAffected` > `alx:hasRange`

This order determines the ordering of the slots Sl in a semantic description. It stipulates a hierarchy over the semantic roles of an n-ary node in the SPARQL graph pattern. It is reflected by an ordering over the syntactic role labels which is roughly equivalent to the linguistic notion of an *obliqueness hierarchy* [13], for example:

`SB` > `OC` > `OC2`

Formally, the obliqueness hierarchy defines a total order $>_L$ over the set of dependency labels L.

For the composition, each of the labels in the label alphabet is assigned one of the semantic operators. The algorithm chooses the operator that is defined to build the composition. The following table shows an excerpt of this mapping:

SB	OC	PNK	MO	PD	PUNC
BIND	BIND	BIND	MERGE	MERGE	IGNORE

The action IGNORE simply skips the interpretation of the subtree. The composition algorithm iterates over the nodes in the dependency graph in a top-down manner, for each edge applying the action defined for the edge label pairwise to each reading of the source and target node.

The algorithm works in a directed manner by sorting the outgoing edges of each node in the dependency graph according to a partial order $\geq_D$.

$$(v_1, l_1, w_1) \geq_D (v_2, l_2, w_2) \Leftrightarrow l_1 > l_2$$

This ordering stipulates a hierarchy over the syntactic arguments in the dependency graph that is reflected by a total ordering on the role labels of the SPARQL pattern graph W: $>_W$. The correspondance between these orders controls the order in which the graph is traversed and therefore, in particular, the correlation of syntactic and semantic roles (semantic role labeling).

The mapping is implemented by the transformation algorithm sketched below. It takes as input a dependency graph $D(V_D, E_D)$ with the root node w_0 as the initial node c in the graph traversation. The nodes are traversed in the order of the hierarchy to assure the correct binding. Note that the transformation may have multiple semantic descriptions as its output. An output semantic description $S = (n, c, Sl, Pr, Fl)$ is only accepted, if all of its slots Sl have been filled. We then apply the toSPARQL operation to arrive at the final SPARQL query.

transform(D,c) : S
$c \leftarrow w_0$: the current node
$S \leftarrow \emptyset$: the set of output readings

```
foreach (c, l, v) in sort(outgoing(c), ≥_D)
begin
    R_c ← readings(c)
    R_v ← transform(v, D)
    foreach (r_c, r_v) in R_c × R_v
    begin
        s ← apply(operator(l), r_c, r_v)
        if(s ≠ NULL)
            S ← S ∪ {s}
    end
end
```

Results

The N-ary modeling requires more triples for simple (binary) facts than using RDF/ OWL properties like DBpedia, because there is always an instance of a relation concept comprising `rdf:type` and participant role triples.

At the time of writing, the Alexandria ontology contained approx. 160 million triples representing more than 7 million entities and more than 13 million relations between them (including literal value facts like amounts, dates, dimensions, etc.). We imported the triples into Virtuoso Open Source Edition, which scales as well as expected with respect to our goals.

80% of all query types understood by the algorithm (i.e. mappable onto valid SPARQL queries) take less than 20ms in average for single threaded linguistic processing on a 64 bit Linux system running on Intel® Xeon® E5420 cores at 2.5GHz, and pure in-memory SPARQL processing by Virtuoso Open Source Edition on a 64 bit Linux system running on eight Intel® Xeon® L5520 cores at 2.3GHz and 32GB of RAM.

The question answering system works fast enough to be used in a multi-user Web frontend like `http://alexandria.neofonie.de`.

The performance of the algorithm is in part due to the high performance of malt parser with a liblinear model, which runs in less than 5ms per question. By using a liblinear model, however, we trade off parsing accuracy against performance in terms of processing time per question. This sometimes becomes noticeable in cases where subject-object order variation in German leads to an erroneous parse.

Answer set	avg. precision	avg. recall	avg. f-measure
All answers	0.25	0.27	0.25
Generated answers	0.49	0.52	0.48
Answers without data mismatch	0.59	0.57	0.57
Generated answers without data mismatch	0.92	0.89	0.89

Table 4. Quality of results of the Alexandria question answering on the QALD-2 training set translated to German.

The performance of the question answering system has been measured using the training set of the QALD-2 challenge[17]. As the question answering in Alexandria currently covers only German, all 100 questions were translated to German first. The results are shown in Table 4. For 49 of the questions no query could be generated. The second row shows the results for the questions for which a SPARQL query could be generated.

It has to be noted that the results provided in the gold standard rely on the DBpedia SPARQL endpoint. As Alexandria is built upon its own schema and the imported data comes from Freebase instead of DBpedia, the comparability of the results is limited. The comparison of both datasets results in various mismatches. For example, the comparison of questions having a set of resources as answer type, is done via the indirection of using the labels. This is possible just because the labels are extracted from Wikipedia by both Freebase and DBpedia. However, some of the labels have been changed during the mapping.

Overall, we have identified the following error types:

1. different labels for the same entities
2. different number of results for aggregate questions
3. query correct but different results
4. training data specifies "out of scope" where we can provide results
5. question out of scope for Alexandria

Type 1 applies particularly often to the labels of movies, most of which are of the form "Minority Report (film)" in DBpedia, and "Minority Report" in Alexandria. Another source for errors (2) results when aggregate questions (involving a count) retrieve a different number of resources. The question "How many films did Hal Roach produce?" for example yields 509 results in DBpedia and 503 results in Alexandria. The third type corresponds to a difference in the data set itself, that is when different information is stored. For example, in Alexandria the highest mountain is the "Mount Everest" whereas in DBpedia it is the "Dotsero".

The last two error types involve questions that are out of scope (4 and 5). The data model used in Alexandria differs from the model in DBpedia as a result to the considerations explained above. On the other hand, Alexandria lacks information since we concentrate on a mapped subset of Freebase. According to the evaluation, the answer "out of scope" is correct if the question cannot be answered using DBpedia.

Out of the 82 questions containing (any) erroneous results 63 belong to one of the error classes mentioned above. The last two rows of Table 4 show the results for all questions that do not belong to any of these error classes.

Acknowledgements

Research partially funded by the German Federal Ministry for Economics and Technology as part of the THESEUS research program[18].

[17] `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=challenge&q=2`

[18] `http://theseus-programm.de/en/about.php`

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia : a crystallization point for the web of data. Web Semantics: Science, Services and Agents on the World Wide Web **7**(3) (2009)
2. Brants, S., Dipper, S., Hansen, S., Lezius, W., Smith, G.: The TIGER Treebank In: Proc. of the Workshop on Treebanks and Linguistic Theories (2002)
3. Cimiano, P.: ORAKEL: A Natural Language Interface to an F-Logic Knowledge Base. In: Proc. of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB) (2004) 401–406
4. Cimiano, P., Haase, P., Heizmann, J., Mantel, M.: Orakel: A portable natural language interface to knowledge bases. Technical Report, University of Karlsruhe (2007)
5. Damljanovic, D., Agatonovic, M., Cunningham, H.: FREyA: an Interactive Way of Querying Linked Data using Natural Language. In: Proc. of QALD-1 at ESWC 2011
6. Elhadad, M., Robin, J.: SURGE: a Comprehensive Plug-in Syntactic Realization Component for Text Generation. Technical Report (1998)
7. Hovy, E.: Methodologies for the Reliable Construction of Ontological Knowledge. In: Proc. of ICCS 2005 91–106
8. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. Journal of the ACM **42** (1995) 741–843
9. Lopez, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: Proc. of ESWC 2006 393–410
10. Lopez, V., Nikolov, A., Sabou, M., Uren, V., Motta, E., DAquin, M.: Scaling Up Question-Answering to Linked Data. In: Cimiano, P., Pinto, H. (eds.): Knowledge Engineering and Management by the Masses. LNCS 6317 (2010) 193–210
11. Nivre, J., Hall, J.: Maltparser: A language-independent system for data-driven dependency parsing. In: Proc. of the 4th Workshop on Treebanks and Linguistic Theories (2005) 13–95
12. Parsons, T.: Events in the Semantics of English: A study in subatomic semantics. MIT Press (1990)
13. Pollard, C., Sag, I.: Information-based Syntax and Semantics, Vol. 1. CSLI Lecture Notes 13 (1987)