From Natural Language Questions to SPARQL Queries: A Pattern-based Approach

Nadine Steinmetz[1] Ann-Katrin Arning[1] Kai-Uwe Sattler[1]

Abstract: Linked Data knowledge bases are valuable sources of knowledge which give insights, reveal facts about various relationships and provide a large amount of metadata in well-structured form. Although the format of semantic information – namely as RDF(S) – is kept simple by representing each fact as a triple of subject, property and object, the access to the knowledge is only available using SPARQL queries on the data. Therefore, Question Answering (QA) systems provide a user-friendly way to access any type of knowledge base and especially for Linked Data sources to get insight into the semantic information. As RDF(S) knowledge bases are usually structured in the same way and provide per se semantic metadata about the contained information, we provide a novel approach that is independent from the underlying knowledge base. Thus, the main contribution of our proposed approach constitutes the simple replaceability of the underlying knowledge base. The algorithm is based on general question and query patterns and only accesses the knowledge base for the actual query generation and execution. This paper presents the proposed approach and an evaluation in comparison to state-of-the-art Linked Data approaches for challenges of QA systems.

1 Introduction

Question answering (QA) is a research discipline at the intersection of natural language processing (NLP), information retrieval, and database processing aiming at answering questions formulated in natural language. Though, QA is a rather old research problem with early system solutions dating back to the sixties, the field has gotten great attention and research made a significant progress over the last few years. This can be exemplified by IBM's DeepQA system Watson which won the quiz show Jeopardy! in 2011. Another well-known example are the personal assistants based on voice recognition such as Apple Siri, Amazon Alexa or Microsoft Cortana which are able not only to execute spoken commands (e.g. "Put x on my shopping list") but also to answer (simple) questions in natural language. However, leveraging large knowledge bases to answer complex questions, supporting question types beyond factoid questions, and dealing with ambiguities are still challenging problems.

QA works usually in a sequence of the following steps: (1) question parsing and focus detection, (2) question classification, (3) query generation, (4) answer candidate generation (query execution), and (5) result ranking. In our work we focus on the steps (3) to (5) where

[1] TU Ilmenau, Databases & Information Systems Group, Ilmenau, Germany, *first.last*@tu-ilmenau.de

doi:10.18420/btw2019-18

we consider structured databases / knowledge bases as sources for answers. Our goal is to provide a *generic* approach not hardcoded for a specific schema or database. For this purpose, we leverage a RDF database such as DBpedia and generate SPARQL queries. A RDF database allows a schema-agnostic approach where the schema has not to be known for query generation because all facts are represented by triples. In addition, it allows to exploit more advanced semantic concepts such as semantic equivalence, similarity or inference mechanisms. Finally, large collections of Linked Data based on a core set such as DBpedia or Wikidata represent a great source for answering a wide range of (not only) factoid questions.

Compared to existing works our query generation approach is independent from the underlying knowledge base by representing queries to answer questions through basic graph patterns whose mapping to knowledge base-specific properties or labels are determined at runtime. Our main contributions are *(i)* a pattern-based approach matching common (but also complex) natural language questions, *(ii)* loosely coupling to an underlying knowledge base, no training or specific information required, and *(iii)* first evaluation results showing similar or even out-performing results compared to specifically trained systems.

2 Problem Statement

The Open Challenge on Question Answering over Linked Data (QALD)[2] has been organized as an evaluation campaign as part of the Extended Semantic Web Conference (ESWC) and the CLEF Initiative (Conference and Labs of the Evaluation Forum) since 2011. The challenge focuses on bringing together scientists who work on question answering and compare new approaches according to a published dataset. The latest challenge – QALD 7 – has taken place at the ESWC 2017. For the evaluation of our approach we will use the training and test datasets provided for this challenge. Evaluation results will be presented in Section 4.

As a first summary the organizers of the QALD challenge published a survey on challenges in Question Answering over Linked Data [Hö17]. The authors give an overview of approaches that have been submitted to several conferences or challenges between 2011 and 2015 – overall they list 72 publications and 62 distinct systems. After review of these systems the authors identified seven challenges developers of QA systems are facing and that addressed in the respective publications. From these challenges, we address the following:

- **Lexical Gap**: As in every QA system the phrases of natural language require to be mapped to parts of the relevant ontology and knowledge base: resources, classes, properties. We try to overcome the lexical gap by creating a set of general questions and retrieve the potentially correct result by matching result type and question type and applying a specific ranking.

[2] `https://qald.sebastianwalter.org/index.php?x=home&q=home`

- **Ambiguity**: The mapping of phrases to the underlying knowledge to retrieve resources, classes and properties often results in a higher amount of output than desired. We apply a scoring at each mapping step and and decide at the end about the correct query containing the correctly mapped phrases by applying also the ranking of the query result.
- **Complex Queries**: SPARQL in its latest version is able to manage queries containing different operators, such as GROUP BY, COUNT, or FILTER operations. We derive the necessity of such operators in a query by identifying the question type and comparing it to the result type of different generated queries.
- **Templates**: For our approach, several question types have been analyzed and we derived respective query transformation patterns that can be applied to any domain or knowledge base w.r.t. the identified question types.
- **Independence from Knowledge Base**: In addition to the ones listed by the authors of [Hö17] we take the challenge of developing a system that works independent from the underlying knowledge base. Our approach is based on general patterns and rankings and knowledge specific lookups.

3 System Architecture

Our presented approach processes given natural language questions in seven different steps, each fulfilling an individual task. The steps are the following (also shown in Figure 1):

1. Question parsing and focus detection
2. Generation of general queries with the phrases of the natural language question according to pre-defined patterns
3. Mapping subject/predicate/object of the general question to representations within the underlying knowledge base
4. Query execution
5. Result ranking
6. Output of the highest ranked SPARQL query and the corresponding result.

Within our algorithm only step 4 and 5 are dependent on the underlying knowledge base as the concrete properties, entities and ontology classes or categories are requested. All other steps are independent from the knowledge base and can be applied to any use case.

3.1 Preliminaries – Knowledge Base Transformation

As described in the previous section, our approach is only loosely dependent on an underlying knowledge base. This means, we are able to work with any knowledge base that fulfills a few a preliminaries:

- the knowledge base is constructed in RDF(S)/OWL

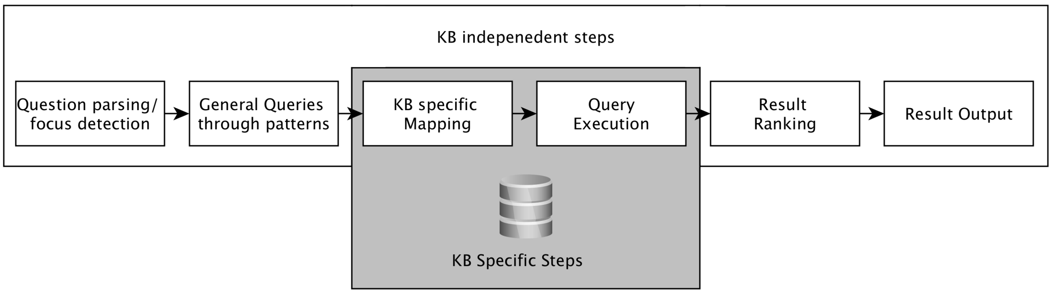


Fig. 1: Overall process of Question Answering

- there is terminological knowledge available about the used vocabulary/ontology (TBox)
- the actual facts are available as assertional knowledge (ABox).

To be able to use the knowledge base, some transformation processes have to be carried out – primarily for reasons of efficient search and lookup. The terminological part of the knowledge base (including OWL and SKOS) is analyzed for class and category labels (for the mapping/lookup process) and transferred to a easily accessible lookup store. Figure 2 shows the RDF(S) parts of a knowledge base which are used to build lookup structures for the knowledge base specific parts of our approach.

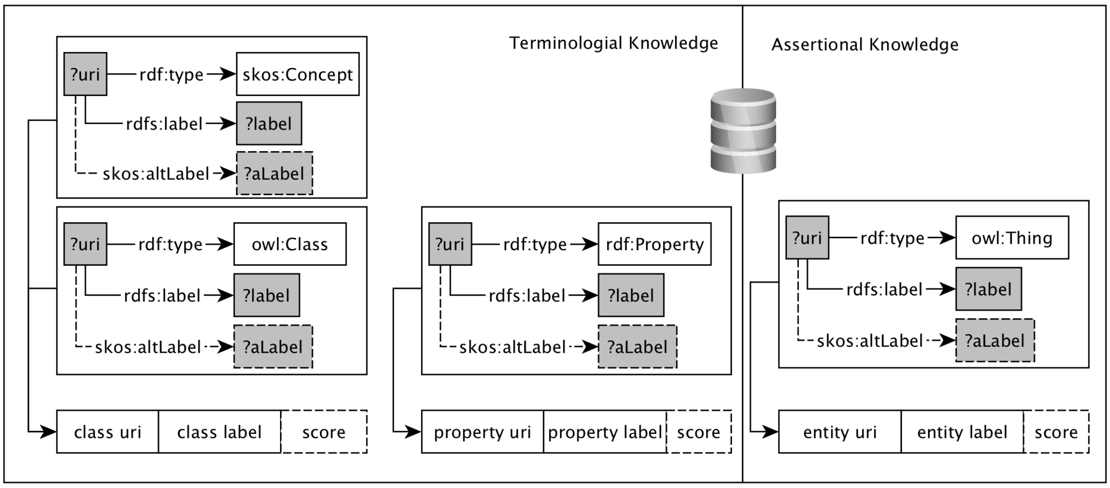


Fig. 2: RDF triples required for generation of lookup structures

We extract the class/category information from the knowledge base to be able to reference type information in questions, such as “Which university is located in Berlin?”. In some cases, the classes included in the ontology do not provide sufficient information about such type information, because the ontology is too general. For instance, for the question “Which Italian dessert contains coffee?” it is required to look up “Italian dessert” within the classes of the ontology. For instance, within DBpedia, the most specific class for this question would be “Food”. But the entity `dbr:Tiramisu`[3] (which would be a correct answer for the question) provides a fact that it is a subject of the category “Italian dessert”. Therefore, we also extract category information provided by `dc:subject`[4] properties in some knowledge bases. The labels for properties are extracted to be able to find verb relationships within the knowledge base, such as a property for relationship between two persons: “Who supervised Alfred Kleiner?”. The labels for the entities are extracted to find the actual subject of a question within the knowledge base.

For these lookup stores (primarily indexed tables in a relational database) the original and – if available – alternative labels of classes/categories, properties (both terminological knowledge) and the actual entities (assertional knowledge) are extracted from the knowledge base.

As we have evaluated our approach based on the dataset provided by the QALD 7 challenge, the underlying knowledge would be DBpedia. For the labels of the entities DBpedia provides more information to be able to collect more synonyms for each entity. As DBpedia is derived from Wikipedia the labels of redirects and disambiguation pages can be used as additional labels. In this way, the entity `dbr:Diana,_Princess_of_Wales` are assigned 25 different labels, e.g. “lady di”, “princess diana”, “lady diana spencer” amongst others. Thereby, the probability of being able to find the correct entity mentioned in natural language is increased. However, the labels of the entities can be more or less relevant. The calculation of a relevance score helps to rank the retrieved entities for a natural language phrase. The same applies for provided alternative labels within the knowledge base (as provided by `skos:altLabel`[5]). Therefore, for each label a score is calculated between [0.0 ... 1.0]. Original labels achieve the highest score. The scores for all alternative labels are calculated according to similarity to the original label, type of the assigned entity (e.g. person’s family names), acronym format etc. The calculation for each score is described more in detail in [St14].

For the mapping of verb phrases from the question to DBpedia properties the ontology only provides original labels and no synonyms, similar to the labels for the ontology classes. This circumstance is an essential disadvantage for the mapping of properties. For instance, the fact that two people are/were married is represented by DBpedia property `spouse` while in natural language several other expressions are used, such as “wife/husband of”, “married”, “in a relationship” etc. Therefore, as an additional source for property labels we are using the PATTY dataset as described in [NWS12] to find potential properties for the verb phrases in the question. The dataset has been derived from a large collection of text documents and we calculated – similar to our entity lookup store – a score for each phrase-property mapping which represents a relevance value for the mapping of the phrase to the property. These scores are used for the ranking of the generated SPARQL queries to find the potentially most correct one for the given input question. As PATTY is a DBpedia specific dataset, our

[3] DBpedia specific prefixes here and in the remainder of the paper stand for: dbr – http://dbpedia.org/resource/; dbo – http://dbpedia.org/ontology/

[4] The Dublin Core vocabulary: `http://dublincore.org/documents/dcmi-terms/`

[5] SKOS Core vocabulary: `https://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/`

approach is restricted to the label extraction as described in Figure 2 when using knowledge bases other than DBpedia.

3.2 Separate Steps of the Algorithm

As depicted in Figure 1 our algorithm consists of separate steps that are either dependent on or independent from the underlying knowledge base. The first parsing and generation of first triples is independent form the knowledge base and can be applied to any vocabulary. Afterwards, the general triples are transformed to knowledge base specific SPARQL queries and are executed on a SPARQL endpoint. In turn, the final ranking and result selection is independent from the knowledge base and builds upon previous ranking and the identified question type. The separate steps are described in detail in the following sections.

3.2.1 Knowledge Base Independent – Question Parsing and Pattern Matching

Question parsing and focus detection. The question is parsed using the Stanford lexical parser[6] [MMM06]. The output of the parser is a parse tree (amongst others) as seen in the example of Listing 1. The parse tree reveals the sentence type as well as the word types (identified as Part-of-Speech (POS) tags) and the relations of the words among one another. In the example the sentence is of type "SBARQ" which means the sentence is "Direct question introduced by a wh-word or a wh-phrase."[7]. To make the sentence type more specific, a second tag is classifying the actual question phrase. In our example it is "WHADVP" which means the question begins with a adverb phrase such as how or why. A full list of identified sentence/question types w.r.t. the dataset used for the evaluation of our system (cf. Section 4) is shown in Table 1. According to the question type the focus of the question is identified. Thereby, the subject that is used as result variable in the SPARQL query is identified. Within the actual question several patterns (combinations of word types) may occur which require to be translated to RDF triples for the SPARQL query. For instance, the phrases "the mayor of Chicago" and "Chicago's mayor" result in different POS tag combinations: NP (the mayor) PP (of) NP (Chicago) and NP (Chicago's) NN (mayor) respectively. Both combinations result in the same RDF triple: `?x onto:mayor res:Chicago .`[8] Such patterns are found in multiple question types and are dissolved independent from the identified question type. Therefore, we identified an extensive list of POS combinations which are translated to RDF triples as described in the following paragraph[9].

List. 1: Sample parse tree for the sentence "When did princess Diana die?"

[6] `https://nlp.stanford.edu/software/lex-parser.shtml`

[7] `https://gist.github.com/nlothian/9240750`

[8] The prefixes here and the reminder of the paper stand for: onto – namespace of the ontology of the underlying knowledge base; res – namespace of the resources of the underlying knowledge base

[9] A list of example sentences, the identified patterns and the transformation to the respective SPARQL query pattern can be found here: `https://bit.ly/2ROwXPM`

```
(ROOT (SBARQ
    (WHADVP (WRB When))
    (SQ (VBD did) (NP (NNP princess) (NNP Diana))
                  (VP (VB die))) (. ?)))
```

Tab. 1: List of identified question types

Sentence Type	Question Type	Example
SBARQ	WHADVP WHADJP WHNP WHPP	When was the Battle of Gettysburg? How much did Pulp Fiction cost? Who designed the Brooklyn Bridge? In which city does the Chile Route 68 end?
S		Show me all books ... . List all basketball players ... .
SQ		Is Berlin the German capital?

Generation of general query triples. The natural language question then requires to be translated to RDF triples. Each RDF triple constitutes a fact which means that the phrases from the question are transferred to single facts. For instance, the question "Show me all books by Joanne K. Rowling." includes two facts:

- the results have to be of type "book", and
- the results are somehow created by "Joanne K. Rowling".

The first fact results in a general triple `?x rdf:type onto:Book .` Here, the object requires to be a class from the underlying ontology. The second fact results in a general triple `?x onto:by res:Joanne_K._Rowling .`[10] Here, the property requires to be included in the ontology and the object requires to be part of the knowledge base. These general triples are generated by analyzing the patterns of POS tags in the parse tree of the question. We identified an extensive list of patterns and assigned respective RDF triples. The respective subject, property and object in the RDF triple are represented by the phrases extracted from the question – as placeholders (except for pre-defined semantic properties, such as the property `rdf:type` which is part of the RDF vocabulary and used as property to state the class membership of a resource of the knowledge base). RDF facts are represented using properties to connect a subject and an object. Due to the fact that our approach is working independent from the underlying knowledge base, we do not know beforehand in which order properties connect subject and object. For instance, for the phrase "supervisor" the knowledge base could contain the property `onto:supervisor` which connects a student in the subject with the respective supervisor in the object. The knowledge base could also contain the property `onto:supervisorOf` which connects the supervisor in the subject with the student in the object. Therefore, we provide two versions for each triple generated from a POS tag combination resulting in more than one SPARQL query for each question. All queries are scored and ranked in a later step.

[10] The triple looks like this before mapping the property and the object to the underlying knowledge base.

3.2.2 Knowledge Base Specific – Mapping and Execution

After the question is parsed and the general triples are generated the underlying knowledge base is taken into account to create the final queries including the knowledge bases specific entities, properties and classes/categories. Therefore, the extracted phrases from the parsed question are replaced URIs and finally the question specific aggregations and other operators are added. The queries are then executed on a SPARQL endpoint.

Subject/property/object mapping. In this step, the generated general queries are transferred to actual RDF triples as specific for the underlying knowledge base.

For the mapping of the subjects of the general triples the extracted phrases from the parse tree are directly looked up. For the objects, three different options are possible:

- direct lookup in case an entity is required according to the pre-defined pattern
- lookup for ontology classes in case a triple for `rdf:type` information is required
- the phrase is inserted as literal.

All mapping processes – for subject, property and object – may result in more than one result per phrase. For each result a query is generated in combination with all already derived queries. Taking into account the mapping scores from PATTY and the entity lookup an overall score for each generated query is calculated by multiplying all derived scores.

The result of this step is a set of RDF triple combinations (as required within the WHERE clause of a SPARQL query) and the corresponding mapping score for each combination. In the next step the final queries are generated and executed.

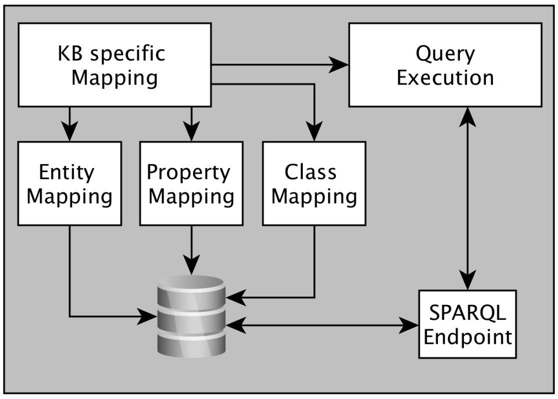
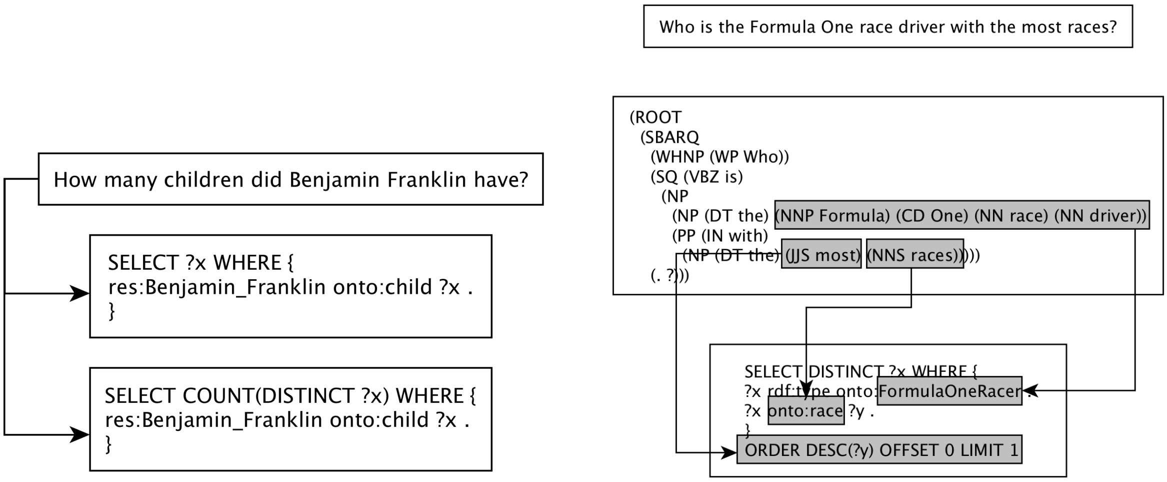


Fig. 3: Components specific for the Underlying Knowledge Base

Query execution. In this step the final queries are created and necessary operators added. The required operators are derived from the question type. For instance, for a WHADJP question – starting with “How many” or “How often” – the expected answer should by a number. As the underlying knowledge base is unknown – as one of the challenges – to our system, the result type "number" can be derived in two different ways: either the range of the mapped property already provides a number or a COUNT operator has to be applied to count the number of resulting entities. Therefore, we create queries for each possibility, such as the example in Figure 4a depicts. In addition to the COUNT operator, other aggregation operators might be applied to the query. For instance, the question "Who is the Formula One race driver with the most races?" requires ORDER, LIMIT and OFFSET operators in the query. According to the POS patterns the required triples are generated and the respective variable used for the ORDER operation is identified. The type of ordering is identified according to pre-defined list of phrases: "most", "highest", "tallest" etc. for descending order or "least", "smallest", "youngest" for ascending order. Figure 4b shows the resulting SPARQL query for the mentioned example. After all queries are created and all required operators are applied the resulting queries are executed on the specific SPARQL endpoint.


How many children did Benjamin Franklin have?
Who is the Formula One race driver with the most races?

```
SELECT ?x WHERE {
res:Benjamin_Franklin onto:child ?x .
}
```

```
SELECT COUNT(DISTINCT ?x) WHERE {
res:Benjamin_Franklin onto:child ?x .
}
```

```
(ROOT
  (SBARQ
    (WHNP (WP Who))
    (SQ (VBZ is)
      (NP
        (NP (DT the) (NNP Formula) (CD One) (NN race) (NN driver))
        (PP (IN with)
          (NP (DT the) (JJS most) (NNS races)))))
    (. ?)))
```

```
SELECT DISTINCT ?x WHERE {
?x rdf:type onto:FormulaOneRacer .
?x onto:race ?y .
}
ORDER DESC(?y) OFFSET 0 LIMIT 1
```

(a) ... for questions where a number is expected as answer type.

(b) ... for questions where ordering of results is required.

Fig. 4: Examples of query generation

3.2.3 Knowledge base Independent – Finalization and Result Output

Result ranking. For each query the results are derived and final ranking is applied according to the expected answer type (as pre-defined from the question type). For most of the question types the answer type is an entity or a list of entities. For question starting with "When" the resulting answer must be a date. And for questions starting with "How many" or "How often" the answer type is a number. The final ranking of the queries and the respective results is applied by comparing the expected answer type and the actual answer type. If the expected and actual answer type are matching the query receives the full score of 1.0. If the query does not produce any result the score naturally is set 0.0. If the query produces a result, but the answer type is not matching to the expected one, the mapping score is set to

0.5. We do not suppress answers with incorrect answer types, because of several reasons. On the one hand, knowledge bases (especially the ones that are automatically extracted or created) some times do not provide any literal types. Dates might not be formatted as date, but still be a correct answer. On the other hand, a question starting with "When"might often require a date as answer type, but a year ("1980") or a season ("Spring 1968") might also be a reasonable answer and contained in the underlying knowledge base like this (instead of a date).

The overall ranking of a query including results is calculated from the mapping score and the answer type score by multiplying both scores. Each query only receives one overall score. The set of results of a query (in case of a list of entities) is regarded as one answer. In this way, all queries are scored and sorted. The query with the highest score is assumed as potentially correct and the respective produced result is set as answer for the input question. At the current stage, our system assumes that a given natural language question actually has an answer. The case of a question that cannot be answered using the underlying knowledge – because of missing information[11] – is not taken into account, yet.

4 Evaluation

Evaluation Dataset. To evaluate our approach a dataset containing the natural language questions as well as the answers formatted as SPARQL endpoint results is required. As stated in [Hö17] only two challenges provide benchmarks with the required structure:

- Open Challenge on Question Answering over Linked Data (QALD) - as introduced in Section 6
- BioASQ – a challenge on large-scale biomedical semantic indexing and question answering[12]

The second challenge only provides questions from the bio-medical domain and the first challenge provides all-purpose QA benchmarks. The DBpedia constitutes the most well-known all-purpose knowledge base provided as RDF. We therefore choose to take into account the QALD challenge and use the dataset of the latest challenge to evaluate our approach[13]. The challenge provides a training dataset some weeks before the submission date and a test dataset to provide the final results of a submitting system. The latest challenge was the first one where participating systems were requested to provide their system as Docker image. In this way, the challenge organizers were able to evaluate the submitted systems directly – in contrast to submitted result files as XML/JSON the years before this last challenge. As the challenge already took place at the Extended Semantic Web Conference (ESWC) we are able to compare our results to the results of the participated systems.

[11] And according to the open world assumption, missing information might mean that something is not existent, but this is not compulsory.

[12] http://bioasq.org/

[13] https://project-hobbit.eu/challenges/qald2017/

The QALD challenge is organized in four tasks. For our evaluation we chose “Task 1: Multilingual question answering over DBpedia” using the English version of the questions. The dataset for this task provides the following information for each question record:

- id – sequential order of the questions
- answertype – either “resource”, “string”, “number”, “date”, or “boolean”
- aggregation – true/false – stating if the required query contains aggregation operators
- onlydbo – true/false – stating if other knowledge bases than DBpedia are required
- question – the actual question provided in eight different languages, for each question additional keywords are provided
- query – a SPARQL query which provides the correct results
- results – the (list of) results – for each result the type and the value are given

For the evaluation of our system we only used the actual natural language question (without the given keywords) and the (list of) results. Everything else (answer type, result type, necessity of aggregation) is not used from the dataset and detected by our system (identified as pattern in the question).

Evaluation Measures. For the comparison of our approach to competing systems the measures Recall, Precision and F_1-Score are used. For each question q recall, precision and F_1-score are calculated as following:

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

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

$$F_1\text{-score} = 2 * \frac{recall(q) * precision(q)}{recall(q) + precision(q)}$$

We calculated all measures as macro measures which means that they have been calculated separately for each question and all measures are averaged over all questions for the overall result.

Evaluation Results. For the QALD 7 challenge only two systems have been submitted providing results for English language: ganswer2 and WDAqua[UNC16]. Thus, we used only these two systems for comparison because only for these systems the results are reproducible. For the calculation of recall, precision and F_1-score only questions have been taken into account for which the system was able to provide a result. We therefore calculated our measures for the evaluation in the same way. The results for the training and test dataset are shown in Table 2. Our approach achieves similar results or even outperforms the other competing system which participated in QALD 7 challenge. For QA systems the precision of the provided answers is more important than the recall: We would rather provide an answer like "Amongst others correct answers are the following" – which means some correct answers might be missing – than proving something "Our answer is this, but it might incorrect". Therefore, our future work will focus on an increased precision – of course in best combined with a reasonable high recall. Nevertheless, these first results show that our approach which is set up to be as independent as possible from the underlying knowledge base is able to compete with other systems that might be trained specifically for a domain and dataset.

Tab. 2: Evaluation results for the QALD 7 training and test datasets

Training Dataset	Recall	Precision	F_1-Score
Our approach	0.588	**0.570**	**0.578**
ganswer2	**0.592**	0.557	0.556
WDAqua	0.540	0.490	0.510
Test Dataset	Recall	Precision	F_1-Score
Our approach	**0.665**	**0.584**	**0.622**
ganswer2	0.498	0.487	0.492
WDAqua	0.160	0.162	0.161

As our approach consists of several separate steps we took a closer look at success and failure of our approach. Therefore, we evaluated each step separately and counted the questions where the step succeeded in comparison to the initial set of questions at the beginning of each step. The results are shown in Table 3. In this overview, two steps respectively the low success rates of two steps are striking: property mapping and final ranking. As described in Section 3.1 we are using the PATTY dataset for the property mapping. Unfortunately, the dataset only contains 58% of all properties contained in the DBpedia ontology. For some reason, very common properties, such as `dbo:deathDate` or `dbo:birthDate`, are missing in the dataset. This fact leads to the low success rate of 52.1% regarding the property mapping. The second lowest success rate is achieved by the overall ranking, which partly is again caused by the PATTY dataset. For each property mapping, a score is provided regarding relevance (cf. Sect. 3.1). Unfortunately, this ranking is sometimes misleading. As this ranking score (together with the scores from subject and property mapping as well as the validity check of the answer type) influences the final ranking, sometimes the wrong query is ranked highest although the used property does not match the phrase of the natural language question at all. The next paragraph will discuss the evaluation results in detail and regarding the proposed challenges.

Discussion. The evaluation results show that our system achieves similar or even better results than the systems that have been submitted to the 7th QALD challenge. However, the detailed analysis of the evaluation results reveals the weakest parts of our approach. As we are using the PATTY dataset to map natural language phrases to properties from the underlying DBpedia knowledge base, there are several questions that cannot be answered, because the dataset does not contain any property for the given phrase. This means, that

Tab. 3: Success rates of separate steps of our algorithm, evaluated on the QALD 7 training dataset

Algorithm step	Evaluation
Overall Questions	166
Focus detection successful	154 / 92.7%
Parsing successful	152 / 98.7%
Triple Generation successful	132 / 86.8%
Subject Mapping successful	127 / 96.2%
Property Mapping successful	62 / 48.8%
Object mapping successful	60 / 96.8%
Query Building successful	60 / 100%
Ranking successful	44 / 73.3%

for many questions no query can be generated, because the respective property is missing. The current DBpedia lists 1439 properties of which only 225 properties are listed in the PATTY dataset. Therefore, only 60% respectively 80% of the questions of the QALD 7 test and training dataset can be answered by our system. Thus, the lexical gap is an essential challenge for QA systems. Another problem constitutes the ranking of queries based on the separate rankings derived from property and subject/object mapping. For the relevance ranking of different properties provided by PATTY we use a co-occurrence measure of phrase and property within the dataset. Unfortunately, this results in some cases in wrong final rankings. For instance, Table 4 shows the rankings of generated queries for the questions “How often did Jane Fonda marry?” and “How often did Alan Rickman marry?”. According to the property relevance ranking the property `dbo:child` achieves a higher score than the property `dbo:spouse`. As all other ranking scores are identical, for the first question the wrong query is chosen as potential correct which delivers the wrong answer. This problem only occurs for entities who actually have children – as it is the case for Jane Fonda. Otherwise – as it is the case for Alan Rickman – the query with the property `dbo:child` does not provide a result (which results in a score of 0.0 for the answer type) and the query containing `dbo:spouse` is ranked highest. These problems result from the structure and quality of the underlying knowledge base respectively the transformed/deployed lookup stores. We discuss this topic further in Section 5. In addition, the evaluation results are influenced by the quality of the datasets. Unfortunately, the datasets contains questions where intuitively the provided answers are wrong. For instance, for the question “Which cities does the Weser flow through?” the entity `dbr:Fulda_(river)` is listed as expected correct result. But this entity is a river and not a city and it results from the missing type check in the SPARQL query provided in the dataset. By adding the type information in the SPARQL query the retrieved results are reduced which results in a lower recall for the evaluation.

Tab. 4: Comparison of Ranking of two queries

Query	Mapping		Answer Type	Final
	Subject	Property		
SELECT (COUNT(DISTINCT ?y) as ?x) WHERE { dbr:Jane_Fonda dbo:child ?y . }	1.0	0.23	1.0	0.614
SELECT (COUNT(DISTINCT ?y) as ?x) WHERE { dbr:Jane_Fonda dbo:spouse ?y . }	1.0	0.08	1.0	0.514
SELECT (COUNT(DISTINCT ?y) as ?x) WHERE { dbr:Alan_Rickman dbo:spouse ?y . }	1.0	0.23	1.0	0.614
SELECT (COUNT(DISTINCT ?y) as ?x) WHERE { dbr:Jane_Rickman dbo:child ?y . }	1.0	0.08	0.0	0.0

5 Challenges

In this section we discuss our experiences during the development of our system especially regarding the challenges described in Section 2.

Lexical Gap. As shown in the evaluation, the lexical gap is an essential factor when natural language questions cannot be answered. For DBpedia, there are multiple sources to extract synonymous labels for the entities, as described in Section 3.1. But for the terminological knowledge – classes/categories and properties – primarily only the original labels are provided. And this is the case for most knowledge bases. Therefore, the quality of the results using other knowledge bases than DBpedia depends on how the lookup stores can be complemented with additional synonymous labels. For this purpose we are following different strategies to be able to include various knowledge bases:

1. determine synonymous information from external vocabularies, such as WordNet[14]
2. extract mappings of natural language phrases and knowledge base properties from embedded RDFa[15] information in text/web documents
3. extract additional information from mapped external ontologies

For the first case, WordNet provides synonymous phrases for each entry within the vocabulary. This information can be mapped to the original labels of the labels of the knowledge base to be complemented. The problem here is that the semantic character of the knowledge base is ignored and synonymous information might be applied to wrong resources. Therefore, this approach requires to be evaluated regarding ambiguity of the knowledge base and quality of the achieved mappings.

For the second strategy, web documents containing RDFa information are analyzed for the deployed terminological and assertional knowledge and thereby the natural language

[14] https://wordnet.princeton.edu/

[15] A markup language extension to HTML5 to enrich web documents with RDF to assign mentioned places, person, etc.: https://www.w3.org/TR/xhtml-rdfa-primer/

information can be mapped to the used vocabulary. Web documents with RDFa annotations are emerging – especially in the field of educational content – and editors have been developed to provide user-friendly interfaces[16]. But the problem is that only a few popular vocabularies and knowledge bases are applied and it might be difficult to find documents containing very specific information as required for QA systems.

The third strategy is the mapping of separate parts of an ontology to other ontologies and get thereby additional information about the mapped parts. Different ontologies might use different labels for equivalent classes or provide additional mappings to further ontologies. Thus, synonymous information might be extracted and complemented in the original ontology.

Ambiguity. Our approach uses scores to rank different generated queries resulting from multiple entities achieved by the mapping process. As shown in Section 4 this separate mapping step is mostly successful (96,2% and 96,8% respectively for subject and object mapping). Therefore, our approach has proven to be successful regarding ambiguity of mapped entities. As discussed in the previous section, the problem of ambiguity is even less difficult for other knowledge bases than DBpedia because of mostly missing synonymous information.

Complex Queries. A natural language question that seems to be simple to be answered might result in a very complex formal query within the QA system. For instance, the question "What is the second highest mountain?" requires the actual triple to find heights of mountains, an operator to sort the heights in descending order, the limit of only one result and an offset to start with the second result in the descending order. All this information needs to be derived from the given natural language phrase. Our approach deduces this information from the question type and specific keywords from the lexically parsed question. The phrases "second" or "third" can be mapped to pre-defined patterns of operators (as described in Section 3.2.2). However, the pre-defined lists of keywords might be dependent on the underlying knowledge base where specific terms and facts might require specific aggregation functions. Therefore, we will explore specific characteristics of domains and topics with the application of our approach to further knowledge bases.

Templates. For the QALD 7 training dataset containing 216 questions we identified six different question types and 36 lexical patterns. Obviously, many lexical patterns are not specific to a question type and occur in various questions. We therefore do not expect the number of required patterns when applying different knowledge bases to increase excessively. It might be necessary to observe knowledge bases more in detail to derive specific patterns. For instance, property-object combinations that occur very often in a knowledge base might

[16] `http://aksw.org/Projects/RDFaCE.html`

be a hint that this is a common fact and might be handled similar to class memberships – without the actual RDF triple of `rdf:type` and the URI of a specific class. For instance, the triple `res:some_City onto:isMetropolis "true".` would be equivalent to a class membership information such as `res:some_City rdf:type onto:Metropolis .` This class membership information must be derived to be able to answer questions like "Which metropolises are located in Europe?". In addition, more general query patterns might be helpful to find the correct fact within the knowledge base. This applies for facts that might be distributed over more than one triple. For instance, within a knowledge base containing information about software projects the question "Who committed the most lines of code?" the information about lines of code might be assigned to a "commit event" including the original committing developer and the facts need to be aggregated over several commit events. This again requires knowledge about the deployed knowledge base which should be extracted during the preprocessing step.

Knowledge Base. Although our approach is set to be as independent from the knowledge base as possible, we are aware of the fact that available information and structure of the data contained in the knowledge base is an essential factor for the quality of a QA system. As described in the previous section, some information might be very domain specific and characteristic for knowledge base and is required to be detected and extracted or complemented before our system could answer specific questions. Another problem might be missing or wrong information in the knowledge base which is often the case for automatically extracted data, as applies for DBpedia. For instance, the information about the elevation of Mount Everest is missing in DBpedia although it is included in the respective Wikipedia article (this fact might be a hint that Siri is using DBpedia for its QA system – cf. Sect. 6). Furthermore, the type information is often missing for entities which leads to missing results when the type is checked in the SPARQL query. Otherwise, too many results might be provided when the type of the questioned subject is omitted. On the other hand, sometimes the type information is set incorrectly for certain entities. For instance, the actor "Terence Hill" is typed as `dbo:Mountain` (strangely along with other persons with the surname "Hill"). This again leads to wrong results when the SPARQL contains the type check. Here again, the solution might be to omit the type check in the query which demands an appropriate query generation algorithm to be able to rule out irrelevant results by additional facts in the query.

6 Related Work

Li et al. introduced an interactive natural language query interface that constructs SQL queries according to natural language questions [LJ14]. The interface therefore interacts with the user and requests feedback on chosen queries respectively concepts. The requesting user is involved in the query construction process at two different stages. First, the natural language phrase is parsed into a lexical tree. Afterwards nodes of the parse tree are identified which can be mapped to components of the requested SQL schema. At this stage the first warning is given back to the user in case the mapping fails for one or more nodes. Also, the successful mappings will be presented to the user. After all nodes are interpreted correctly (with the help of the user) the linguistic parse tree is adapted to be valid for their system. Implicit nodes can be inserted, if necessary, and the user is able to support this process. The verified query tree is then transferred to a SQL statement containing joins, aggregate functions etc.

Deutch et al. presented an similar approach where a natural language query is transferred to a lexical query tree and the nodes are mapped to variables to be able to construct a conjunctive query (CQ) [DFG17]. The authors not only focus on the conversion process to be able to query a database for the requested natural language question, but also provide a natural language answer according to provenance tracking of tuples from the query tree and applying them to the answer tree.

Amongst others these two approaches present interfaces to convert natural language to SQL statements. In contrast to that our approach is built upon RDF knowledge bases respectively constructing SPARQL queries and does not use any user feedback for creating the formal queries. The following approaches are therefore all based on RDF knowledge bases and focus on translating natural language (NL) questions to SPARQL queries.

Freitas et al. developed a vocabulary independent approach for querying Linked Data so the user does not need to know about the vocabulary of the data sets. The main components of their system are (1) entity search, (2) spreading activation and (3) measuring semantic relatedness [Fr11]. The challenges their system is able to deal with are the lexical gap, ambiguity, complex operators and distributed knowledge. The essential question behind the key entity search is which components of the natural language query of the user can be mapped to classes or instances in the knowledge base. After detecting those key entities, the URI of the key entity is looked up and used as a pivot entity for the next step, which will be spreading activation. Before spreading activation can be started, the NL query has to be parsed in a form that can be mapped to the SPO form of data represented in RDF. After that, every pivot entity is used as a starting point for the spreading activation process. If there is more than one node that is higher than a given relatedness threshold, it will lead to a second path and in the end the path with the highest relatedness wins.

The system PARALEX introduced by Fader et al. in 2013 faces the challenges lexical gap, ambiguity as well as procedural, temporal or spatial questions and uses templates similar to our approach [FZE13]. However, PARALEX has a slightly different approach with focussing on a broad range of different questions asking for the same fact rather than on complex questions. It is only able to handle queries that can be represented as a triple, for example `portrayer(Emma Watson, Hermione Granger)`. PARALEX tries to map a given question to be answered by such a triple. Examples for those questions are: “Who portrayed Hermione Granger?”, “Who is the actress of Hermione Granger?”, “Who played the role of Hermione Granger?” or “Who is the actress that played the role of Hermione Granger in

Harry Potter?”. Although those questions may differ, they can all be answered by the factoid triple above. The system stores triple entries in a database that are of the form `r(e1, e2)` and it can answer queries with one unknown variable that is either `e1` or `e2`. Therefore queries look like `r(?, e2)` for example for the question “Who portrayed Hermione Granger?” or `r(e1, ?)`, which would answer the question “Which roles did Emma Watson play?” if we assume a triple like `portrayed(Emma Watson, Hermione Granger)` alongside others. To map the phrases of a question to an entry in the triple data store, a lexicon had been built by means of a learning algorithm which consists of two stages: first it is initially set up and then derivates are added to increase the precision. The main disadvantage of the system is the missing ability to answer complex questions, because it is not able to build the query.

In 2014, Xu et al. introduced their system Xser [Xu14] and participated with their approach in the QALD 5 challenge (held at CLEF 2015 Initiative) achieving the highest recall and precision (recall=0.72, precision=0.74) among the competing systems. Their approach is based on two steps: first the NL question is analyzed for predicate argument structures using an semantic parser. In the second step, the actual queries based on the underlying knowledge base are generated. In contrast to our system, their approach requires to be trained using knowledge base specific training data.

Bast et al. presented an extensive survey in the field of semantic search on text and knowledge bases [BBH16]. Within this survey several combinations of the search input and the underlying data are explored. Obviously, a semantic search can be performed by analyzing a natural language as input. Therefore, the authors examine several approaches on understanding natural language and finding relevant search items for the input query.

Also in 2015, the system SemaGraphQA introduced by Beaumont et al. participated in the QALD 5 challenge [BGL15], achieving significantly worse results compared to the best performing system Xser as previously introduced (recall=0.32, precision=0.31). They participated again in the QALD 6 challenge achieving an increased precision but decreased recall (recall=0.25, precision=0.70). Their system uses a graph-based approach matching parts of the NL question to the knowledge base and building a syntactic graph.

The best performing system of the QALD 6 challenge (co-located with ESWC 2016) has been CANaLI introduced by Mazzeo et al. [MZ16]. CANaLI achieved a overall precision and recall of 0.89 for english questions of Task 4 of the challenge (“Multilingual question answering over DBpedia”). However, the system is working on a semi-automatic basis using user feedback to map to correct properties and entities in the knowledge base. We also use the datasets of Task 4 of QALD-6 to evaluate our system, but will not compare to CANaLI, because our system is working completely self-sufficient from the external help of the user.

When it comes to explaining the research field of question answering to people outside of the research community popular systems like Alexa, Siri & Co. perfectly fit to describe the challenges. These systems seem to know and understand everything and the research field might appear redundant. But, a closer look reveals the limits of the systems. The domain of understanding a user is often very limited. For instance, you can ask for tomorrow's weather or where some place is or to call somebody from your contact list (using your smartphone). Additionally, some simple questions can be answered: The question "How high is Mount McKinley?" is correctly answered with "Denali is 20,308 ft above sea level" (even replacing the name with current correct one). But the similar question "How high is Mount Everest?" only answered by providing the article about Mount Everest on Wikipedia. The more complicated example (from the QALD challenge) "Who became president after JFK died?" only gives web resources regarding John F. Kennedy. These examples show that these systems are limited to popular and rather simple questions. The foundation of these systems are trained databases containing e.g. often requested web queries and the corresponding most frequent chosen web result.

7 Conclusion

In this paper, we presented our novel approach to question answering over Linked Data knowledge bases. In addition to the common challenges of QA systems, we designed our system to be as independent as possible from the underlying knowledge base. The preliminary transformation process extracts required information from the RDF(S) knowledge base and this information is used for a mapping step as part of the complete algorithm. Afterwards a SPARQL endpoint is required to execute the actual queries. A long-term goal is to provide a web interface where a knowledge base can be uploaded by a user, the transformation process is conducted and the SPARQL endpoint established. Subsequently, the user is able to ask natural language questions on the prepared knowledge base. Our approach has been evaluated on the latest datasets provided by the QALD challenge. First results show that our approach performs similarly or even better than the competing systems that participated in the last challenge. Our future work includes two main aspects: on the one hand we want to achieve a reasonable saturation of patterns for general purpose questions. This means, we will further explore common natural language questions for so far missing patterns without overfitting the assembled set of patterns and be as general as possible. On the other hand, we will transfer our approach to knowledge from other domains and further research the applicability of the detected patterns so far. Overall, we were able to show that a QA system is not necessarily required to be trained on a specific domain or knowledge base. Nevertheless, the discussed challenges provide sufficient input for further enhancements and developments in this complex research field.

8 Acknowledgements

This work was partially funded by the German Research Foundation (DFG) under grant no. SA782/26.

References

[BBH16] Bast, Hannah; Björn, Buchhold; Haussmann, Elmar: Semantic Search on Text and Knowledge Bases. Found. Trends Inf. Retr., 10(2-3):119–271, June 2016.

[BGL15] Beaumont, R.; Grau, B.; Ligozat, A.-L.: SemGraphQA@QALD5: LIMSI participation at QALD5@CLEF. In: Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015. 2015.

[DFG17] Deutch, D.; Frost, N.; Gilad, A.: Provenance for Natural Language Queries. Proc. VLDB Endow., 10(5):577–588, 2017.

[Fr11] Freitas, A.; Oliveira, J.G.; O'Riain, S.; Curry, E.; Da Silva, J.C.P.: Querying Linked Data Using Semantic Relatedness: A Vocabulary Independent Approach. In: Proc. of the 16th Int. Conf. on Natural Language Processing and Information Systems. NLDB'11. Springer-Verlag, pp. 40–51, 2011.

[FZE13] Fader, A.; Zettlemoyer, L.; Etzioni, O.: Paraphrase-driven learning for open question answering. In: Long Papers, volume 1. Association for Computational Linguistics (ACL), pp. 1608–1618, 2013.

[Hö17] Höffner, K.; Walter, S.; Marx, E.; Usbeck, R.; Lehmann, J.; Ngonga Ngomo, A.-C.: Survey on Challenges of Question Answering in the Semantic Web. Semantic Web Journal, 8(6), 2017.

[LJ14] Li, F.; Jagadish, H. V.: Constructing an Interactive Natural Language Interface for Relational Databases. Proc. VLDB Endow., 8(1):73–84, September 2014.

[MMM06] Marneffe, M.; Maccartney, B.; Manning, C.: Generating Typed Dependency Parses from Phrase Structure Parses. In: Proc. of the 5th Int. Conf. on Language Resources and Evaluation (LREC-2006). European Language Resources Association (ELRA), Genoa, Italy, May 2006.

[MZ16] Mazzeo, G. M.; Zaniolo, C.: Answering Controlled Natural Language Questions on RDF Knowledge Bases. In: Proc. of the 19th Int. Conf. on Extending Database Technology, EDBT 2016, Bordeaux, France. pp. 608–611, 2016.

[NWS12] Nakashole, N.; Weikum, G.; Suchanek, F.: PATTY: A Taxonomy of Relational Patterns with Semantic Types. In: Proc of the 2012 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. EMNLP-CoNLL '12, pp. 1135–1145, 2012.

[St14] Steinmetz, N.: Context-aware semantic analysis of video metadata. Phd. thesis, Universität Potsdam, 2014.

[UNC16] Unger, C.; Ngonga Ngomo, A.-C.; Cabrio, E.: 6th Open Challenge on Question Answering over Linked Data (QALD-6). In: Semantic Web Challenges: Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers. pp. 171–177, 2016.

[Xu14] Xu, K.; Zhang, S.; Feng, Y.; Zhao, D.: Answering Natural Language Questions via Phrasal Semantic Parsing. In: Proc. Natural Language Processing and Chinese Computing: Third CCF Conference (NLPCC), Shenzhen, China. Springer, pp. 333–344, 2014.