An introduction to Question Answering over Linked Data

Christina Unger and André Freitas and Philipp Cimiano

Abstract. While the amount of knowledge available as linked data grows, so does the need for providing end users with access to this knowledge. Especially question answering systems are receiving much interest, as they provide intuitive access to data via natural language and shield end users from technical aspects related to data modelling, vocabularies and query languages. This tutorial gives an introduction to the rapidly developing field of question answering over linked data. It gives an overview of the main challenges involved in the interpretation of a user's information need expressed in natural language with respect to the data that is queried. The paper summarizes the main existing approaches and systems including available tools and resources, benchmarks and evaluation campaigns. Finally, it lists the open topics that will keep question answering over linked data an exciting area of research in the years to come.

1 Introduction

The amount of structured knowledge available on the web is growing steadily. The linked data cloud, consisting of a large amount of interlinked RDF (*Resource Description Framework*[1]) datasets, now comprises more than 30 billion RDF triples[2]. Knowledge bases such as Freebase[3] and DBpedia[4] are huge and become more and more popular for various applications. Structured data is by now also collected and exploited by search engines such as Google, e.g. in the form of knowledge graphs[5] that are used to enhance search results.

As the amount of available structured knowledge keeps growing, intuitive and effective paradigms for accessing and querying this knowledge become more and more important. Over the past years, there has been a growing amount of research on interaction paradigms that allow end users to profit from the expressive power of Semantic Web standards while at the same time hiding their complexity behind an intuitive and easy-to-use interface. Especially natural language interfaces such as question answering systems have received wide attention [29], as they allow users to express arbitrarily complex information needs in an intuitive fashion and, at least in principle, in their own language. In contrast to traditional search engines, question answering systems allow users to pose a (possibly complex) full fledged question, instead of merely a list of keywords, and return precise answers, instead of documents in which the answer can be

[1] `http://www.w3.org/TR/rdf-primer/`
[2] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`
[3] `http://www.freebase.com/`
[4] `http://dbpedia.org/`
[5] `http://www.google.com/insidesearch/features/search/knowledge.html`

potentially found. Prominent examples of question answering systems are Wolfram Alpha[6] and IBM's Watson[7], which won the game show Jeopardy! in 2011 against two of the best human players.

Originally, question answering had a strong focus on textual data sources to find answers, relying mostly on information retrieval techniques. In the early 70's, question answering then started to incorporate structured data, developing natural language interfaces to databases [1]. Nowadays, with the growing amount of knowledge in the linked open data cloud, interest in question answering over structured data is quickly regaining interest.

The key challenge for question answering over linked data is to translate the users' information needs into a form such that they can be evaluated using standard Semantic Web query processing and inferencing techniques. Over the past years, a range of approaches have been developed to address this challenge, showing significant advances towards answering natural language questions with respect to large, heterogeneous sets of structured data. In this tutorial, we give an introduction to this exciting, growing field of research.

We start with an overview of the challenges involved in answering questions over linked data in Section 2. Then, Section 3 provides the anatomy of a typical question answering system over linked data, presenting the components that most systems implement in one way or another. Equipped with this general architecture, Section 4 summarizes some of the prominent approaches to question answering over linked data, describing existing systems that are representative for these approaches in more detail. In Section 5 we then list tools and resources that proved useful for question answering over linked data and that can get you started in building your own system. Next, Section 6 mentions measures for evaluating question answering systems and points to some important benchmarking campaigns. Finally, Section 7 reviews the open topics and challenges for future research.

Throughout the tutorial we assume that you have a working knowledge of RDF and its query language SPARQL. For an introduction or refresher, we recommend the W3C primers `http://www.w3.org/TR/rdf11-primer/` and `http://www.w3.org/TR/rdf-sparql-query/`.

2 Main challenges

If a common web end user wanted to access information in the linked data cloud, he would face two obstacles. First, the amount of available datasets is huge and it is by no means trivial to identify and find those datasets that contain the information he is looking for. Second, once he found a relevant dataset, he would need to formulate a query that retrieves the information, e.g. in SPARQL[8], the standard query language for RDF data. To this end, he needs to speak SPARQL

[6] `https://www.wolframalpha.com/`

[7] `http://researcher.ibm.com/researcher/view_project.php?id=2099`

[8] `http://www.w3.org/TR/rdf-sparql-query/`

and he needs to know the vocabulary and schema underlying the dataset he wants to query.

Since the common web user is usually not familiar with Semantic Web languages or the structure of the linked data cloud and the available datasets, question answering systems aim to bridge the gap between the user and the (structure of the) data, by translating between an information need expressed in natural language on the one hand side and structured queries and answers on the other hand side. In the following, we describe the major challenges in doing so.

2.1 Bridging the gap between natural language and linked data

The major task for question answering systems is to interpret the user's information need expressed in natural language with respect to the data that is queried. Consider a simple example: With respect to DBpedia, the question 1a can be expressed by means of the SPARQL query 1b:[9]

1. (a) What is the currency of the Czech Republic?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
    res:Czech_Republic  dbo:currency  ?uri  .
}
```

In order to get from the question to the query, we need to know that the name the Czech Republic corresponds to the resource `res:Czech_Republic`, that the expression currency corresponds to the property `dbo:currency`, and we need to know the structure of the query, i.e. that the entity `res:Czech_Republic` is the subject of the property and that the object is to be returned as answer.

While constructing the SPARQL query from the question is (relatively) straightforward in this particular example, very often the process is much more involved. In most cases it involves two challenges: mapping natural language expressions to the vocabulary elements used by the data, accounting for lexical and structural mismatches in doing so, and handling meaning variations introduced by ambiguous and vague expressions, anaphoric expressions, and so on. Let us look at both challenges in turn.

Mapping natural language expressions to vocabulary elements URIs are language-independent identifiers. Although they usually bear mnemonic names, their only actual connection to natural language is by the labels that are attached to them. These labels often provide a canonical way to refer to the URI, but usually do not account for lexical variation. The class `dbo:Film`, for example, has

[9] Throughout the tutorial, we will use the following prefixes: `dbo` for `http://dbpedia.org/ontology/`, `dbp` for `http://dbpedia.org/property/`, and `res` for `http://dbpedia.org/resource/`.

the English label film but does not capture other variants such as movie. Similarly, the property dbo:spouse bears the English label spouse, while natural language knows a wide varierty of ways of expressing this relationship, among them wife of, husband of, and to be married to, which are more likely to occur in a user question than the somehwat more formal term spouse.

So although the vocabulary of natural language and the vocabulary used by the data overlap, the expressions a user uses often differ from the labels attached to the data. Bridging the resulting *lexical gap* is thus one of the challenges that a question answering system needs to address. The following example 2 illustrates vocabulary similarities and differences.

2. (a) Which Greek cities have more than 1 million inhabitants?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
     ?uri rdf:type dbo:City .
     ?uri dbo:country res:Greece .
     ?uri dbo:populationTotal ?p .
     FILTER (?p > 1000000)
}
```

It is more or less straightforward to match the expression cities to the class dbo:City having the label city. Less straightforward is matching have inhabitants to the property populationTotal; here the similarity between both exists only on the semantic level but not on the string level. Furthermore, Greek needs to be matched with the property dbo:country with fixed object Greece. This already points to another difficulty: differences in the structure of the question and the query.

Structural differences are due to the fact that the conceptual granularity of language does often not coincide with that of the schema underlying a particular dataset. On the one hand side it can be that natural language is more granular than the data, as in the following example 3, where the structure of the natural language questions suggests a relation join that relates two entities, Germany and the EU, while the required property is dbp:accessioneudate, relating a country to the date when it joined the EU.

3. (a) When did Germany join the EU?

   (b)
```
SELECT DISTINCT ?date
WHERE {
     res:Germany dbp:accessioneudate ?date .
}
```

On the other hand side it can be that the data is more granular than natural language. In the following example 4, there is one natural language expressions great-grandchildren that corresponds to a property chain consisting of three times the property dbo:child.

4. (a) Who are the great-grandchildren of Bruce Lee?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
    res:Bruce_Lee dbo:child ?c1 .
              ?c1 dbo:child ?c2 .
              ?c2 dbo:child ?uri .
}
```

In addition to mapping natural language expressions to vocabulary elements underlying a particular dataset, there are expressions that do not correspond to any vocabulary element but rather have a fixed, dataset-independent meaning. Examples are quantifiers like the most (see example 5), comparative expressions like more than (see example 6) and less than, cardinals and superlatives (see example 7). These expressions correspond to aggregation operations in SPARQL, such as filtering, ordering and limits.

5. (a) Who produced the most films?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
    ?x rdf:type dbo:Film .
    ?x dbo:producer ?uri .
}
ORDER BY DESC(COUNT(?x))
LIMIT 1
```

6. (a) Which cities have more than three universities?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
    ?x rdf:type dbo:University .
    ?x dbo:city ?uri .
}
HAVING (COUNT(?x) > 3)
```

7. (a) What is the second highest mountain on Earth?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
        ?uri rdf:type dbo:Mountain .
        ?uri dbo:elevation ?x .
}
ORDER BY DESC(?x)
OFFSET 1 LIMIT 1
```

Meaning variation Question answering systems involve process natural language and thus inherit the challenges involved in processing natural language in general. On of these challenges is dealing with ambiguities. Ambiguity covers all cases in which a natural language expression can have more than one meaning, in our case can map to more than one vocabulary element in the target dataset. For instance, different vocabularies usually offer different ways of answering an information need. In 8, having adopted the Euro can be expressed either by means of the property `dbo:currency` with the object resource `res:Euro`, or by means of the property `dbp:currency` with the object literal `'EUR'`. In this case, both mappings are appropriate; we thus constructed a query taking the union of the results of both mappings.

8. (a) Which countries adopted the Euro?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
    ?uri rdf:type dbo:Country .
  { ?uri dbo:currency res:Euro . }
  UNION
  { ?uri dbp:currencyCode 'EUR'@en . }
}
```

In other cases only one mapping is appropriate, often depending on the context. For example, to retrieve the mayor of a city from DBpedia, the corresponding property is `dbo:mayor` in the case of Lyon, `dbo:leader` in the case of Berlin, and `dbo:leaderName` in the case of Tel Aviv.

A more extreme form of ambiguity arises from semantically light expressions, such as the verbs to be and to have, and prepositions like of, with, etc. What vocabulary element they need to be mapped to strongly depends on the linguistic context they occur in. Consider the verb to have. In 9, have corresponds to the property `dbo:exhibits`, wile in 10, have corresponds to the property `dbo:location`. The only difference between both questions is that the former is about museums and painting, and the latter is about countries and caves.

9. (a) Which museum has the most paintings?

   (b)
```
SELECT DISTINCT ?uri
WHERE {
    ?uri rdf:type dbo:Museum .
    ?x   rdf:type dbo:Painting .
    ?uri dbo:exhibits ?x .
}
ORDER BY DESC(COUNT(?x))
LIMIT 1
```

10. (a) Which country has the most caves?

    (b)
```
SELECT DISTINCT ?uri
WHERE {
    ?uri rdf:type dbo:Country .
    ?x   rdf:type dbo:Cave .
    ?x   dbo:location ?uri .
}
ORDER BY DESC(COUNT(?x))
LIMIT 1
```

Of course, semantically light expressions can in one context also be mapped to different vocabulary elements, as the preposition in in the following example.

11. (a) Give me all companies in Munich.

    (b)
```
SELECT DISTINCT ?uri
WHERE {
    ?uri rdf:type dbo:Company .
  { ?uri dbo:location res:Munich . }
  UNION
  { ?uri dbo:headquarter res:Munich . }
  UNION
  { ?uri dbo:locationCity res:Munich . }
}
```

2.2 Multilinguality

Multilinguality has become an issue of major interest for the Semantic Web community, as both the number of actors creating and publishing data all in languages other than English, as well as the amount of users that access this data and speak native languages other than English is growing substantially. In order to achieve the goal that users from all countries have access to the same information, there is an impending need for systems that can help in overcoming language barriers by facilitating multilingual access to semantic data originally produced for a different culture and language.

In principle, the Semantic Web is very well suited for multilinguality, as URIs are language-independent identifiers. However, in order to access and use these identifiers in different language contexts, it is important to have labels in these languages. But adding multilingual labels is not common practice. A recent study [21] has shown that the majority of datasets is monolingual: Less than a quarter of the RDF literals have language tags, and most of those tags are in English.

In the context of question answering over linked data, a challenge is to interpret questions in multiple languages. For example, all the questions in 12 taken from the QALD-4 benchmarking dataset express the same information need and thus should be mapped to the same SPARQL query 13.

12.
- *English:* Which German cities have more than 250000 inhabitants?
- *German:* Welche deutschen Städte haben mehr als 250000 Einwohner?
- *Spanish:* ¿Qué ciudades alemanas tienen más de 250000 habitantes?
- *Italian:* Quali città tedesche hanno più di 250000 abitanti?
- *French:* Quelles villes allemandes ont plus de 250000 habitants?
- *Dutch:* Welke Duitse steden hebben meer dan 250000 inwoners?
- *Romanian:* Ce orașe germane au mai mult de 250000 de locuitori?

13.
```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT DISTINCT ?uri
WHERE {
       { ?uri rdf:type dbo:City . }
       UNION
       { ?uri rdf:type dbo:Town . }
         ?uri dbo:country res:Germany .
         ?uri dbo:populationTotal ?population .
         FILTER ( ?population > 250000 ) }
```

2.3 Data quality and data heterogeneity

Strong requirements on question answering systems are completeness and accuracy (wrong answers are often worse than no answers). In the context of linked data, this especially requires a system to deal with heterogeneous and imperfect data. First of all, the knowledge available on the linked data cloud is incomplete, so question answering systems should be able to detect when the queried data sources do not contain the answer. Second, the datasets sometimes contain duplicate information, with different datasets using different vocabularies even when talking about the same things. Possibly, datasets also contain conflicting information and inconsistencies, which have to be dealt with.

2.4 Performance and scalability

The temporal performance of question answering systems over linked data is rarely reported in the literature. In [27], FREyA is reported to require 36 seconds on average to answer questions over DBpedia, while Aqualog reports an average of 20 seconds on the same dataset, and these are figures for one dataset only. This shows that yielding performant systems that provide timely answers is challenging. The challenge here is to cope with large datasets comprising billions of triples. Performance in real time, which we regard as processing a question in under a second, can only be obtained by using appropriate index structures, search heuristics and possibly adopting distributed computing principles. Also, systems are often considered scalable only if the response time is not be proportional to the size of data being accessed.

2.5 Coping with distributed and linked datasets

Only few systems yet address the fact that the structured data available nowadays is distributed among a large collection of interconnected datasets, and that answers to questions can often only be provided if information from several sources are combined. In fact, the issue of how to evaluate natural language questions of distributed but linked datasets has not been investigated extensively yet. Exceptions are PowerAqua (see Section 4.1) and the system by Ngonga et al. [33], which builds on federation and attempts to decompose a question into several parts that can be answered with respect to different datasets.

Consider the example query *What are side effects of drugs used for the treatment of Tuberculosis?* (taken in modified form from [33]), which can be answered using three different (linked) datasets, i.e. the Sider dataset[10] (containing information about drugs and their side effects), the Diseasome dataset[11] (containing information about diseases and genes associated with these diseases) and Drugbank [12] (containing comprehensive knowledge base containing information about drugs, drug target (i.e. protein) information, interactions and enzymes, etc.).

2.6 Integrating structured and unstructured data

In addition, a lot of information is still available only in textual form, both on the web and in the form of labels and abstracts in linked data sources. Therefore approaches are needed that can not only deal with the specific character of structured data but also with finding information in several sources, processing both structured and unstructured information, and combining such gathered information into one answer.

An example for a question requiring both structured and unstructured data when being answered over DBpedia is the following one. Here, the information that needs to be extracted from free text is marked by the prefix `text`.

14. (a) Where did the first man in space die?

    (b)
```
SELECT DISTINCT ?uri
WHERE {
        ?x text:"is" text:"first man in space" .
        ?x dbo:deathPlace ?uri .
}
```

Identifying the person that was the first man in space is possible only by means of the free text abstract, while his death place is encoded only in the RDF data.

Approaches that exploit both structured and unstructured data are very rare so far. A notable exception is the system by Fader et al. [14].

3 The anatomy of a question answering system

All approaches to question answering over linked data share the challenges mentioned in the previous section, and most of the systems rely on the same kind of components to address them, although they often differ in their particular implementation. This section introduces the subtasks involved in the overall task of answering questions, and then presents the general architecture of a prototypical question answering system over linked data, explaining the components that most existing systems employ.

[10] http://sideeffects.embl.de/
[11] http://diseasome.kobic.re.kr/
[12] http://www.drugbank.ca/

3.1 Dimensions of question answering

Question answering is a complex and multi-dimensional task. Categorizing the different dimensions of question answering is fundamental for understanding the different phenomena and challenges involved in a question answering task. Additionally, the categorization helps in the definition of the scope of a question answering system and to set up a proper evaluation. This section describes the most important dimensions involved in question answering and their associated elements, including types of questions and answers, data source types, and key functionalities.

Question and answer type Question classification is often done based on a categorization of the answers that expected, based either on their *form* or their *type*. Classifying questions with respect to the answer form results in a question taxonomy roughly along the following lines:

- *Factoid questions*, including
    - *Predicative questions*, e.g.
        - Who was the first man in space?
        - What is the highest mountain in Korea?
        - How far is it from Earth to Mars?
        - When did the Jurassic Period end?
        - Where is Taj Mahal?
    - *List questions*, e.g.
        - Give me all cities in Germany.
    - *Yes/No questions*, e.g.
        - Was Margaret Thatcher a chemist?
- *Definition questions*, e.g.
    - Who was Tom Jobim?
- *Evaluative or comparative questions*, e.g.
    - What is the difference between impressionism and expressionism?
- *Association questions*, e.g.
    - What is the connection between Barack Obama and Indonesia?
- *Explanation/Justification questions*, e.g.
    - Why did the revenue of IBM drop?
- *Process questions*, e.g.
    - How do I make a cheese cake?
- *Opinion question*, e.g.
    - What do most Americans think of gun control?

Question answering systems often focus on factoid and definition questions. In the context of the linked data cloud this is mainly due to this being the kind of information represented in the available datasets.

Classifying questions with respect to the type of the expected answer leads to answer taxonomies such as the one proposed by Li & Roth [25]. The following list gives their five high-level categories together with examples of the more fine-grained subcategories:

- *Abbreviation*
- *Entity:* event, color, animal, plant,. . .
- *Description:* definition, manner, reason,. . .
- *Human:* group, individual,. . .
- *Location:* city, country, mountain,. . .
- *Numeric:* count, date, distance, size,. . .

In addition to the form or type of answer, a more fine-grained question classification is possible by means of *question focus* and *topic*, representing what the question is about. For example, in the question What is the height of Mount Everest?, the focus is the property height, and the topic is, more generally, geography. In the question What is the best height increasing drug?, the focus looks similar, while the topic is medicine.

Data sources Question answering systems differ with respect to the data source(s) they are able to process to derive an answer. On the one hand side, they usually consume a specific type of data:

- *Structured data*, e.g. relational databases and linked data
- *Semi-structured data*, e.g. XML documents
- *Unstructured data*, e.g. text documents

Hybrid question answering systems are able to process a combination of two or more of the types of data mentioned above.

On the other hand side, question answering systems differ with respect to the number of data sources they consider:

- a *single* dataset
- an enumerated list of *multiple*, distributed datasets
- all datasets available on a *large scale*, i.e. considering all datasets of a certain kind (e.g. structured or text) available on the web or in the linked data cloud

Furthermore, a question answering system can be either *domain-specific*, addressing very specific knowledge in a particular domain, e.g. the biomedical or financial domain, or *open-domain*, addressing general knowledge, especially encylopaedic knowledge, common-sense knowledge, news or trivia.

Finally, the modality of the data that is considered is often text or structured knowledge, but can also include other modalities such as images, sound, and video.

Complexity of the question answering task The complextiy of the question answering task can be classified according to different dimension involved in interpreting the question and processing the data, such as the following ones:

- *Semantic tractability* [31], specifying the closeness between the natural language question and the formal query or answer in terms of lexical and structural similarity

– *Semantic complexity*, comprising the level of complexity of the domain, the amount of ambiguity and vagueness in the question, as well as data heterogeneity
– *Answer locality*, specifying whether the answer is wholly contained in one dataset, or snippets from which the answer can be composed are distributed across different datasets
– *Derivability*, specifying whether the answer is explicit or implicit, in the latter case, e.g., requiring additional reasoning

3.2 Components of a question answering system

Despite the architectural particularities of different question answering systems, there are high-level functionalities and components which are present in most of them. These components are depicted in Figure 1 and described in the following.

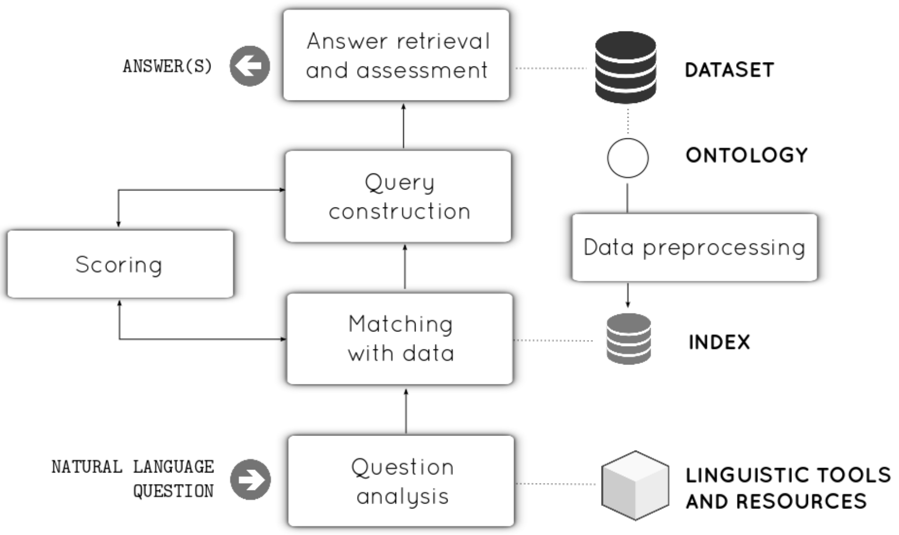


Fig. 1. High-level components of question answering systems over linked data.

Data preprocessing: Preprocessing the information present in a dataset helps to reduce the runtime of a system. For example, systems often rely on an index of the dataset in order to match natural language expressions with labels of vocabulary elements. This also speeds up processing.

Question analysis: The first step of a question answering system often consists in the linguistic analysis of the question, as well as the detection and extraction of question features. The linguistic analysis can involve both a syntactic and a semantic analysis, commonly relying on tools such as part-of-speech taggers and parsers, on resources such as dictionaries, e.g. WordNet or Wiktionary, and possibly some logical representation formalism to capture the meaning of a question. It also involves steps like Named Entity Recognition. In addition, a question can be analysed with respect to the categories mentioned above, in particular detecting the question type, the focus and the expected answer type.

Data matching: In order to deal with the lexical and structural differences between the question vocabulary and the dataset elements, a component is required that matches question terms with the dataset terminology. It can range from simple term look-up to more sophisticated semantic matching strategies.

Query construction: Based on the results of the first two components, the input question is transformed into a structured query.

Scoring: Usually, both the data matching and the query construction components output several candidates that need to be scored and ranked. Criteria for scoring can include different similarity measures (e.g. string similarity and semantic relatedness), popularity of data elements (e.g. based on their frequency), and coherence of the candidates and their combinations with the data schema.

Answer retrieval and assessment: Next, the constructed structured query needs to executed over the database, extracting the answer to the input question. This component possibly also comprises an assessment of the answers, e.g. checking whether the type of the answer fits the expected answer type.

Especially if several datasets are queried that provide either different parts of an answer or overlapping and conflicting answers, the scoring and integration of answers becomes crucial, taking into account information provenance and reliability, i.e. how trust-worthy the source is from which the answer was extracted.

Answer presentation: Finally, the answer needs to be presented to the user. Especially for question answering systems over linked data aiming at making structured data accessible for common end users not familiar with Semantic Web languages, it is important to not return answers as URIs or triples but in a more comprehensible format, e.g. in natural language or as a visual graph.

4 Overview of state-of-the-art systems

There is a range of approaches to meeting the challenges we described in Section 2. They vary in several aspects, e.g. being domain-specific or schema-agnostic, relying on a deep linguistic analysis, on statistical methods, or on graph exploration algorithms, and involving different resources. In general, approaches to question answering over linked data can be classified into the following types of systems.

Approaches based on controlled natural languages Approaches based on controlled natural languages, e.g. GiNSENG [3], typically consider a well-defined restricted subset of natural language that can be unambiguously interpreted by a given system.

Approaches based on formal grammars: Systems based on formal grammars, such as ORAKEL and Pythia (see Section 4.2 below), typically rely on linguistic grammars that assign a syntactic and semantic representation to lexical units, and exploit the principle of compositional semantics to compute an overall semantic representation of a question by combining the meaning of the parts as specified in the grammar. The benefit of such systems are that, if the corresponding constructs are covered by the grammar, they can deal with questions of arbitrary complexity. The clear drawback is their brittleness, as they fail if a certain question can not be parsed by the grammar.

Approaches based on mapping linguistic structures to ontology-compliant semantic structures: Systems such as Aqualog and PowerAqua (see Section 4.1 below) in contrast adopt a more shallow strategy and try to directly match linguistic structures to semantic triples. To this end, they rely on a measure of similarity that determines the similarity between elements in the query and predicates, subjects or objects in the knowledge base, more or less aiming at computing a bijective mapping between the elements of the query and resources or predicates. QAKIS [5], for instance, tries to establish a match between fragments in the natural language question and textual patterns that were automatically collected from Wikipedia. Other systems along these lines are FREyA [10] and Querix [22]. While such approaches are more robust than approaches based on formal grammars, they suffer from the fact that there needs to be a one-to-one correspondence between the syntactic structure of the question and the semantic representation. An approach aims to remedy this is [39], which generates relaxed query variants that cover the user question to different extents, especially leaving out intractable parts and instead adding them as textual conditions that are then used in determining mappings from natural language expressions to URIs and for disambiguation.

Template-based approaches Template-based approaches, such as LODQA and TBSL (see Section 4.3 below) implement a two-stage process for transforming a natural language question into a SPARQL query. First, they construct a template (or pseudo-query) on the basis of a linguistic analysis of the input question. Second, this template is instantiated by matching the natural language expressions occuring in the question with elements form the queried dataset. The main weakness of such approaches is that often the templates closely correspond to the linguistic structure of the question, thus failing in cases of structural mismatches between natural language and the dataset. While structural variations can be included in the templates, this usually explodes the number of possible queries to build and score.

Graph exploration approaches Graph exploration approaches, such as Treo (see Section 4.4 below), Top-k exploration [34] and the approach by Ngonga et al. [33], interpret a natural language question by mapping elements of the question to entities from the knowledge base, and then proceeding from these pivot elements to navigate the graph, seeking to connect the entities to yield a connected query. The main bottleneck of graph-based approaches is that an exhaustive search of the graph is unfeasible, so that approaches typically explore the graph up to a certain depth or implement some heuristics to make the search over the graph more efficient. This is particularly relevant when the data is not available locally but needs to be explored via HTTP-requests, as is the case for linked data. Treo for example implements a heuristic search over linked data, relying on spreading activation guided by a measure of semantic relatedness. Another drawback of graph-based approaches is that more complex queries involving aggregation, e.g. *Which researcher has written the most papers in the Semantic Web area?* cannot be answered as they do not have a one-to-one correspondence to a path in the graph.

Machine learning approaches A number of recent approaches consider question answering as a machine learning problem. Examples are [32], who propose a model for joint query interpretation and response ranking, [2] and [24], who aim at learning semantic parsers given a knowledge base and a set of question/answer pairs, and [6], who combine a standard supervised algorithm for learning semantic parsers with an algorithm for matching natural language expressions and ontology concepts, and an algorithm for storing matches in a parse lexicon.

In the following, we will discuss a subset of the above mentioned systems in more detail, in particular PowerAqua, one of the first question answering systems for RDF knowledge bases, our own systems Pythia, TBSL, and Treo, as well as the DeepQA approach developed by IBM Watson.

4.1 Aqualog and PowerAqua: Querying on a Semantic Web scale

PowerAqua [26] is a question answering system focusing on querying multiple datasets on the Semantic Web. PowerAqua is an evolution of the AquaLog system [28], one of the first question answering systems targeting Semantic Web data.

Figure 2 depicts the high-level architecture of the PowerAqua system, described in more detail in the following. The first step consists in a linguistic

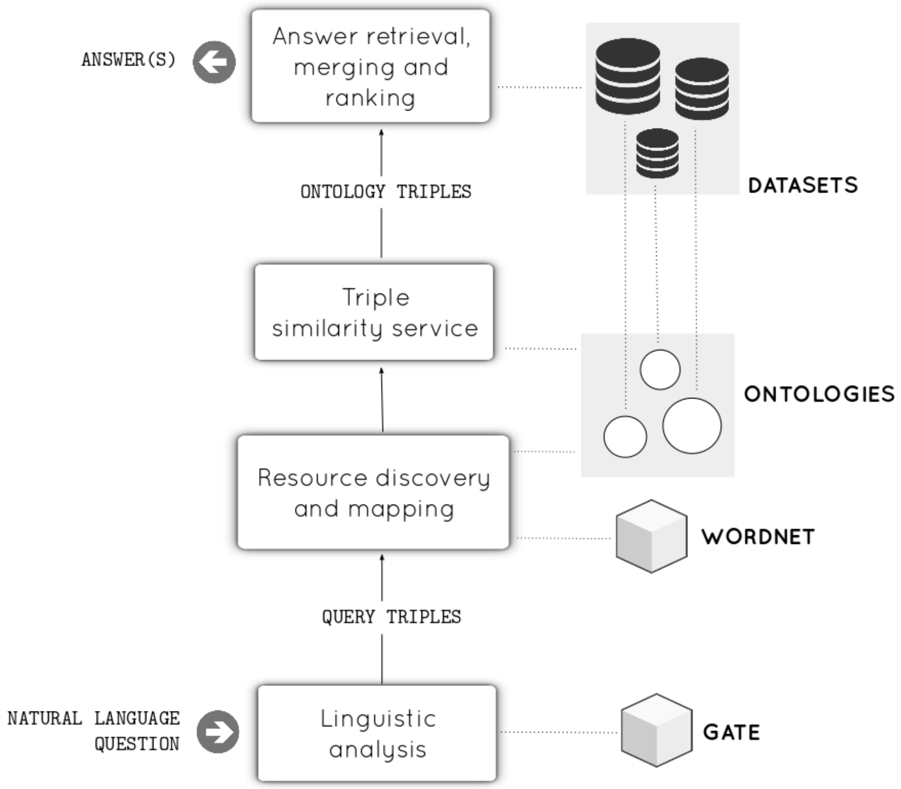

ANSWER(S)
Answer retrieval, merging and ranking
DATASETS
ONTOLOGY TRIPLES
Triple similarity service
ONTOLOGIES
Resource discovery and mapping
WORDNET
QUERY TRIPLES
NATURAL LANGUAGE QUESTION
Linguistic analysis
GATE

Fig. 2. Architecture of PowerAqua

analysis of the question using GATE[13], in order to detect the question type and to translate the natural language question into a triple-based representation, into so-called *query triples*. Here are two examples of question and the corresponding query triples that are generated:

15. Give me actors starring in movies directed by Clint Eastwood.
    - ⟨actors, starring, movies⟩
    - ⟨actors/movies, directed, Clint Eastwood⟩
16. Find me university cities in Japan.
    - ⟨university cities, ?, Japan⟩

The subject of the second triple in 15, actors/movies, represents an ambiguity with respect to which of both terms fulfil the role. The property in 16, on the other hand, is unknown, as it stems from the semantically light preposition in.

[13] `https://gate.ac.uk/`

The second step then searches for candidate instantiations of the occurring terms. This involves detecting ontologies on the Semantic Web that are likely to contain the information requested in the question, and finding vocabulary elements that match the terms. Matching natural language terms and vocabulary elements relies both on string-based similarity measures, allowing for exact matches (`Japan`) and approximate matchings (`CountryJapan`), and on WordNet synonyms, hypernyms and hyponyms as well as on `owl:sameAs` links. Moreover, this component uses word sense disambiguation techniques to disambiguate different interpretations of the question terms across ontologies. The output is a set of tables containing matching semantic elements for each term occurring in the query triples. These mappings can be used to turn query triples into *ontology triples*. For example, given the mappings found in DBpedia for the terms in the first query triple in 15, this triple can be transformed into the following ontology triples:

– ⟨`Actor`, `starring`, `Film`⟩
– ⟨`Actor`, starring, `American_movie`⟩

Next, the relation similarity service (RSS) component determines the most likely interpretation both of the terms in the query triples and the question as a whole, on the basis of the linguistic analysis of the question and the ontological context. This can also involve modifying query triples, e.g. by splitting compound terms. The term university cities, for example, can be split, yielding a new query triple:

17. ⟨cities/universities, ?, Japan⟩

Finally, given a ranked list of ontology triples, answers are retrieved from the involved data sources. Since these can be multiple different sources, this step also involves identifying semantically equivalent or overlapping information in the answer set, in order to avoid duplicate results, a ranking of answers, as well as integrating answer fragments from different sources.

PowerAqua's main strength is that it locates and integrates information from different, heterogeneous semantic resources, relying on query disambiguation, and ranking and fusion of answers. Its main weakaness, on the other hand, is that due to limitations in GATE it cannot cope with aggregation, i.e. questions involving counting (e.g. how many), comparisons (such as higher than or more than), and superlatives (such as the highest and the most).

4.2 ORAKEL and Pythia: Ontology-specific question answering

ORAKEL [9] and Pythia [37] are ontology-based question answering systems. This means that ontologies play a central role in interpreting user questions. For example, ontological knowledge is used for drawing inferences, e.g. in order to resolve ambiguities or to interpret semantically light expressions. But most importantly, user questions are interpreted with respect to an ontology underlying the dataset that is queried. Unlike systems that first construct general, dataset-independent meaning representations (like the triple representations built by PowerAqua) and only subsequently try to match this with the target data, ORAKEL and Pythia construct meaning representations whose vocabulary is already aligned to the vocabulary of the ontology. To this end, they rely on ontology-lexica that make the possible linguistic realizations of ontology concepts in a particular language explicit, e.g. in *lemon* [30] format. *lemon* is a model for the declarative specification of multilingual, machine-readable lexica in RDF that capture syntactic and semantic aspects of lexical items relative to some ontology. The meaning of a lexical item is given by reference to an ontology element, i.e. a class, property or individual, thereby ensuring a clean separation between the ontological and lexical layer. Since *lemon* abstracts from specific linguistic theories and grammar formalisms, all linguistic categories such as part of speeches, syntactic roles and frames have to be defined in an external linguistic ontology, such as *LexInfo* [8].

Both ORAKEL and Pythia implement the architecture depicted in Figure 3. First, the ontology lexicon is used to automatically construct principled linguis-

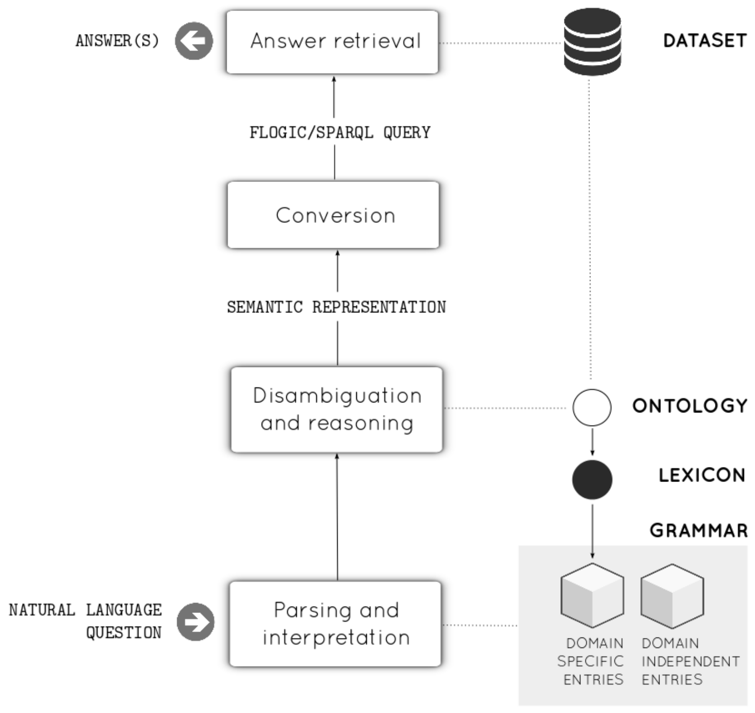
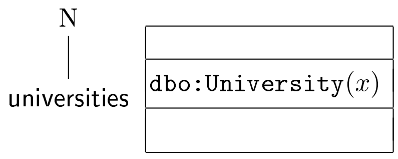


Fig. 3. Architecture of ORAKEL and Pythia

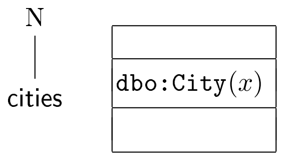
tic representations. ORAKEL relies on *Logical Description Grammars* (LDG) as syntactic formalism and an extended version of lambda calculus for specifying semantic representations; Pythia builds on *Lexicalized Tree Adjoining Grammars* [35] (LTAG) as syntactic formalism and *Dependency-based Underspecified Discourse Representation Structures* [7] (DUDES) for specifying semantic representations. Those linguistic representations – together with domain-independent representations, e.g. for determiners and auxiliary verbs – constitute the grammar that is used for parsing and interpreting an input question. Syntactic and semantic analysis work in parallel. In particular, the interpretation process is compositional, meaning that the semantic representation of the input is recursively computed on the basis of the meaning of the words in the input as well as the way the words are connected syntactically. Finally, the resulting meaning representations are transformed into formal queries, in particular F-Logic and SPARQL queries.

For instance, for the question Which cities have more than three universities?, the following processing would take place. First of all, ORAKEL and Pythia require an ontology-lexicon. This lexicon would specify the English lexicalizations university and alma mater for the ontology class `University`, the lexicalization city for the ontology class `City`, and, for example, the lexicalization located in for the ontology property `city`. On the basis of these lexicalization, syntactic and semantic representations are constructed. Here we only show the ones that are relevant for the interpretation of the questions:

18. a.



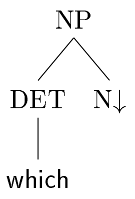
b.



The meaning representations can be viewed as first-order logic like representation. Note that they already use the vocabulary of the ontology.

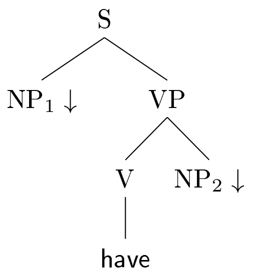
In addition, the grammar contains entries for domain-independent expressions, in particular which, more than, three, and to have:

19. a.



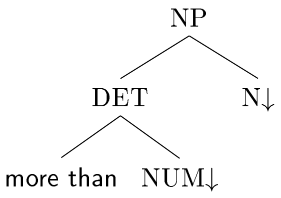
$?x$
(x, N)

b.



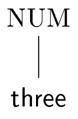
$P(y, x)$
(x, NP_1), (y, NP_1)

c.



x
$\texttt{COUNT}(x) > n$
(x, N), (n, NUM)

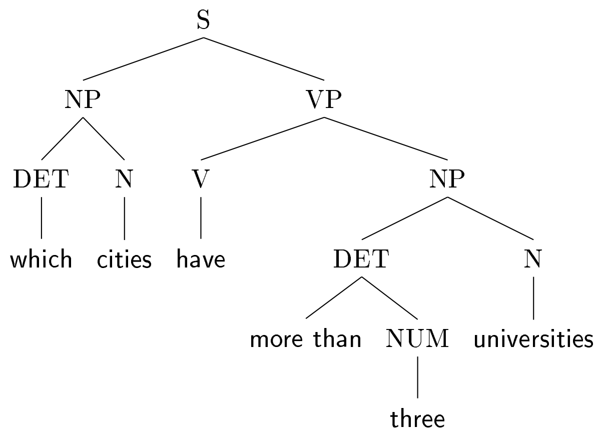
d.



n
$n = 3$

Tree nodes marked with $\downarrow$ constitute so-called substitution nodes, which can be replaced by a tree having a root node of the specified category. The meaning representations consist of three parts: a set of variables (where variables marked with a question mark will be returned by the final SPARQL query), a set of first-order logic like conditions, and a set of pairs specifying the role that the meaning contributions of the arguments play (e.g. (x, N) means that the meaning of the N node will be unified with x). The meaning repesentation of the semantically light verb to have moreover contains a predicate variable P, which will need to be resolved with respect to the ontology vocabulary later. Both the lexicon and the generated domain-specific grammar entries as well as the domain-independent grammar entries are constructed offline and then used during the run-time of the systems. Using the trees in 18 and 19 for parsing yields the tree shown in 20a. In parallel to the syntactic analysis, the semantic representations in 18 and 19 are combined into the resulting meaning given in 20b.

20. (a)



(b)

$$\begin{array}{|l|}\hline ?x\ y\ n \\ \hline \texttt{dbo:City}(\mathrm{x}) \\ \texttt{dbo:University}(\mathrm{y}) \\ P(y,x) \\ \texttt{COUNT}(x) > n \\ n = 3 \\ \hline \\ \hline \end{array}$$

In order to resolve P, the disambiguation and reasoning steps retrieves all those properties from the ontology that are compatible with domain `University` and range `City`. In the case of DBpedia, these are the properties `city`, `location`, and a few more. At this point, Pythia would assume that all of them are valid instantiations of P, and it would retrieve answers for all possible interpretations. So one of the queries resulting from converting the meaning representation of the question into SPARQL would be the following one:

21.
```
SELECT DISTINCT ?x
WHERE {
  ?x rdf:type dbo:City .
  ?y rdf:type dbo:University .
  ?y dbo:city ?x .
}
GROUP BY ?y
HAVING (COUNT(?y) > 3)
```

Implementing a principled, deep linguistic analysis allows ORAKEL and Pythia to construct formal queries even for complex natural language questions, e.g. involving quantification and superlatives. Moreover, since the resulting meaning representations are built from atomic building blocks that were generated from an ontology lexicon, they use a vocabulary that is aligned to the vocabulary of the ontology. This ensures a precise and correct mapping of natural language terms to corresponding ontology concepts. The use of an ontology lexicon therefore offers a very precise way of matching natural expressions with ontology concepts.

ORAKEL and Pythia thus are in this sense question answering systems specific for a given ontology that require an ontology lexicon for the ontology that models the data that is queried. The bottleneck here consists in the effort required for building such lexica, either manually or semi-automatically [38]. And although a grammar-based interpretation process offers high precision, systems relying on domain grammars usually suffer from limited coverage.

4.3 TBSL and LODQA: Template-based question answering

TBSL [36] and LODQA [23] are template-based approaches that rely on a parse of the user question to produce a query template. This template mirrors the linguistic structure of the question and is, in a second step, instantiated by mapping the occurring natural language expressions to the domain vocabulary. The main intuition is that the linguistic structure of a question together with expressions like more than and the most already determine part of the query that is needed in order to retrieve answers. This part is domain-independent and provides a skeleton structure of the query that then needs to be filled in with domain-specific vocabulary elements.

More specifically, TBSL implements the architecture depicted in Figure 4. For linguistic analysis, TBSL relies on a parsing and interpretation step just like Pythia (described above). The only difference is that no domain-specific lexicon is required. Instead, the input question is first processed by a part-of-speech tagger. The resulting part-of-speech tags are used to create general linguistic representations on the fly. These representations are underspecified in the sense that their meaning with respect to the queried data is still unknown. Like in the case of Pythia, these representations are, together with pre-defined domain-independent representations, used for parsing, which leads to an underspecified semantic representation of the natural language question (still without specific domain vocabulary elements) that is then converted into a SPARQL query template. This conversion relies on heuristic rules specifying that verbs usually correspond to properties, that nouns and adjectives correspond to classes or properties, and that proper names correspond to resources.

The query templates that TBSL builds for the question Who produced the most films, for example, are given in 22. They consist of a SPARQL query containing slots for all missing elements that need to be filled based on the domain vocabulary. The slots contain information about the type of missing element (class, property or resource) and the natural language expressions it should correspond to. The latter are the words that are contained in the input question together with synonyms retrieved from WordNet.

22. (a)
```
SELECT DISTINCT ?x
WHERE {
  ?y rdf:type ?c .
  ?y ?p ?x .
```

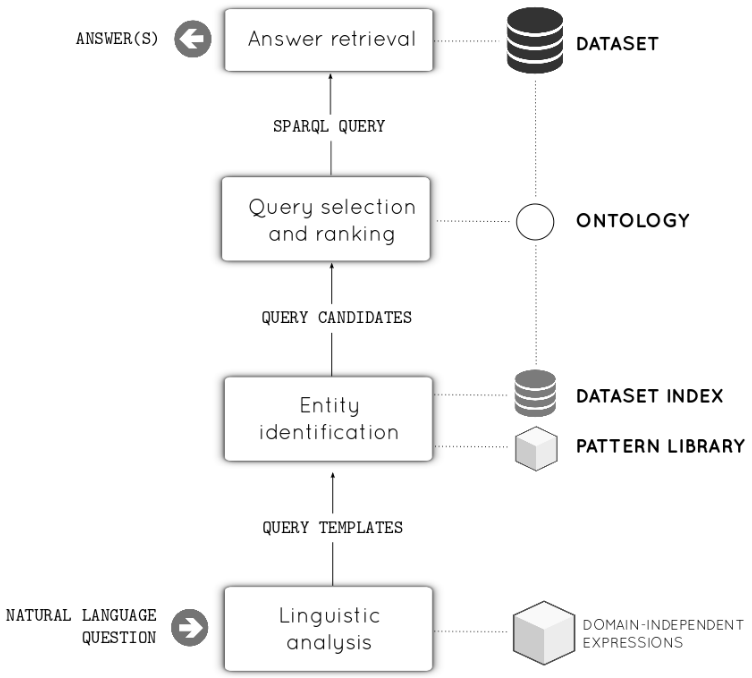


Fig. 4. Architecture of TBSL

```
}
ORDER BY DESC(COUNT(?y))
LIMIT 1
```

– (?c,Class,[films])
– (?p,Property,[produced])

```
SELECT DISTINCT ?x
WHERE {
  ?x ?p ?y .
}
ORDER BY DESC(COUNT(?y))
LIMIT 1
```

(b) – (?p,Property,[films])

In order to obtain URIs that fill the slots, TBSL uses an index to look up entities that match with the required class and given natural language expressions. The matching relies on string similarity, matching with WordNet synonyms, and on an existing collection of natural language patterns, BOA [19]. The basic idea behind BOA is to use the instance knowledge available in a dataset to compute natural language expressions that stand for properties from the underlying ontology. To this end, for any given property, BOA first retrieves pairs of those entities that the property relates. The labels of those entities are then used to search a text corpus for sentences in which they occur, from which possible natural language verbalizations of the property are extracted. Similarly, an ontology-lexicon as used by ORAKEL and Pythia could be employed as pattern collection.

The result of the entity identification step is a list of URI candidates for filling the slots, ranked regarding their string similarity values and prominence values (i.e. their frequency in the dataset). For our example question and considering DBpedia as dataset, candidates for filling the class slot would include the classes `Film` and `FilmFestival`, and candidates for filling the property slot would include the properties `producer` and `wineProduced`, among a range of others.

These candidates are used to construct all possible query instantiations, representing potential translations of the input question. These queries are ranked regarding the rank of the entities occuring in them. Moreover, TBSL performs a schema conformance check, in particular checking whether the types of the occuring entities are compatible with the domain and range of those properties that relate the entities. For example, the class candiates `Film` and `FilmFestival` are incomaptible with the property candidate `wineProduced`, as the domain of the latter is `WineRegion`.

Finally, the highest ranked queries are tested against the underlying triple store and the best answer is returned to the user. For the question Who produced the most films, for example, the highest ranked query is the following one (with a score of 0.76).

23.
```
SELECT DISTINCT ?x
WHERE {
?x <http://dbpedia.org/ontology/producer> ?y .
?y rdf:type <http://dbpedia.org/ontology/Film> .
}
ORDER BY DESC(COUNT(?y))
LIMIT 1
```

An enhancement of TBSL and LODQA is the question answering system platform developed in the context of the OKBQA hackathon[14] in 2014, open and freely accessible on GitHub: `https://github.com/okbqa`. Some of the main improvements concern the template generation component, which relies on a more robust linguistic analysis using dependency parsing and semantic role labeling instead of grammars. Also, the main problem that TBSL faces is that the constructed query templates are too fixed. In order to cover all possibe template structures as well as all possibilities of filling the slots of those template with all possible entity candidates results in huge candidate query sets. Instead it is desirable to not fix the triple structure of the query body but rather determine it by means of dataset exploration, e.g. relation finding.

[14] `http://www.okbqa.org`

4.4 Treo: Schema-agnostic querying using distributional semantics

Addressing the vocabulary mismatch problem for databases is central to querying large schema and heterogeneous linked datasets. One of the main challenges in addressing the vocabulary mismatch problem is its dependency on large-scale common-sense or domain-specific knowledge. As an example, suppose we want to interpret the question Is Chelsea Clinton married? to the associated data (`:Chelsea_Clinton :spouse :Marc_Mezvinsky`). Mapping the query to the data depends on semantically matching married to spouse. Representing common-sense knowledge using structured large-scale knowledge bases comes with the price of data acquisition (manually or through automated information extraction methods), data representation and reasoning over large-scale knowledge bases. These are on their own right major challenges in artificial intelligence research. Existing structured resources such as WordNet do not fully address the vocabulary problem [16].

The Treo approach focuses on addressing this challenge using *distributional semantic models*. Distributional semantics is defined upon the assumption that the context surrounding a given word in a text provides important information about its meaning [?]. Distributional semantics focuses on the automatic construction of a semantic model based on the statistical distribution of word co-occurrence in texts, allowing the creation of an associational and quantitative model which captures the degree of semantic relatedness between words. Distributional semantic models are represented by Vector Space Models (VSMs), where the meaning of a word is represented by a weighted vector which captures the patterns of co-occurrence with other words in the corpus. Distributional models focus on complementing approaches such as WordNet and ontology-based approaches, trading structure for volume of commonsense knowledge [18] and automatic construction capabilities. In the Treo system, a distributional semantic model is used as a core element to address the query-dataset vocabulary gap together with a compositional model based query patterns.

Figure 5 depicts the high-level workflow behind Treo using the question Who is the daughter of Bill Clinton married to? as example. The first step (step 1) is the construction of a distributional semantic model based on the extraction of co-occurrence patterns from large corpora, which defines a distributional semantic vector space. The distributional semantic vector space uses concept vectors to semantically represent data and queries, by mapping datasets elements and query terms to concepts in the distributional space. Once the space is built, the RDF graph data is embedded into the space (step 2), defining the $\tau - Space$, a structured distributional semantic vector space. The alignment between structured data and the distributional model allows the use of the large-scale common-sense information embedded in the distributional model (extracted from text) to be used in the *semantic matching/approximation* process.

After the data is indexed into the $\tau - Space$, it is ready to be queried. The query processing starts with the analysis of the natural language question, from which a set of query features and a semi-structured query representation is extracted (step 3). After the query is analyzed, a *query processing plan* is

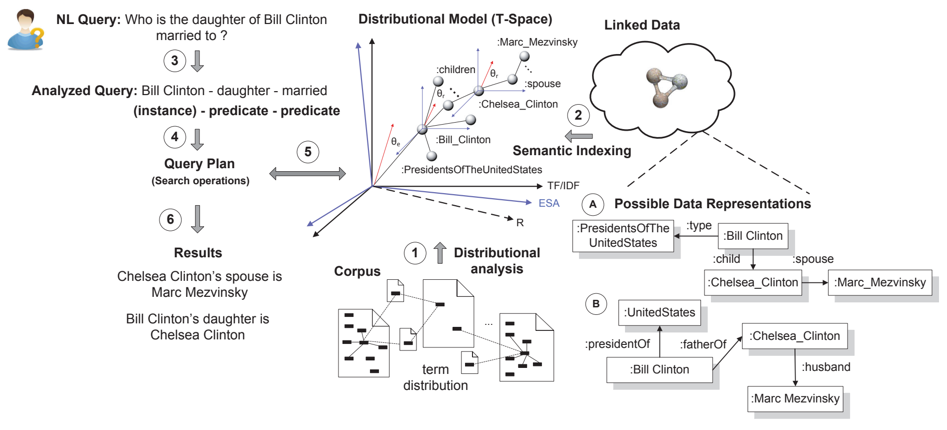

NL Query: Who is the daughter of Bill Clinton married to ?
3
Analyzed Query: Bill Clinton - daughter - married
(instance) - predicate - predicate
4
Query Plan
(Search operations)
5
6
Results
Chelsea Clinton's spouse is Marc Mezvinsky
Bill Clinton's daughter is Chelsea Clinton
Distributional Model (T-Space)
:Marc_Mezvinsky
:children
:spouse
:Chelsea_Clinton
:Bill_Clinton
:PresidentsOfTheUnitedStates
TF/IDF
ESA
R
Linked Data
2
Semantic Indexing
1
Distributional analysis
Corpus
term distribution
A
Possible Data Representations
:PresidentsOfThe UnitedStates
:type
:Bill Clinton
:child
:spouse
:Chelsea_Clinton
:Marc_Mezvinsky
B
:UnitedStates
:presidentOf
:fatherOf
:Chelsea_Clinton
:Bill Clinton
:husband
:Marc Mezvinsky

Fig. 5. Example of the vocabulary gap between query and data representation

generated, which maps the set of features and the semi-structured query into a set of search, navigation and transformation operations (step 5) over the data graph embedded in the $\tau-Space$. These operations define the semantic matching between the query and the data, using the distributional semantic information. This corresponds to the compositional model associated to the distributional model.

Question analysis The question analysis step consists in recognizing and classifying entities and operations in the question, as well as in mapping the natural language question into a *partial and ordered dependency structure* (PODS), a triple-like pattern, and a set of query features, see Figure **??**. The specific question analysis operations are the following ones.

– **Question parsing** consists in the parsing of the question according to its dependency structure and the occurring parts of speech, see Figure 6.
– **Question/answer feature detection and classification** consists in the detection of the query focus and answer type based on rules over part-of-speech tags.
– **Entity detection and classification** uses part-of-speech tag pattern rules to determine the type of detected entity candidates (instances, classes and properties). Examples of those rules ares the following ones:
  - NNP+ $\rightarrow$ Instance
  - {RB* JJ*} NN(S)* {IN NNP}* $\rightarrow$ Class OR Property
  - BE* VB {IN NN}* $\rightarrow$ Property
– **Operation detection** uses part-of-speech tags and keyword patterns to detect operations and associated parameters in the question. While the lexical and structural variation for dataset elements is large, the vocabulary for typical database operations can be enumerated in a knowledge base of lexical expressions of operations *Op*.

- **Triple-pattern ordering** reduces the dependency structure constructed when parsing the input question to a set of PODS by applying two sets of operations: the removal of stopwords and their associated dependencies, and the re-ordering of the dependencies based on the core entity position in the query (where the core entity becomes the first query term and the topological relations given by the dependencies are preserved). For the example question, the PODS is Bill Clinton – daughter – married to. A detailed description of the triple-pattern processing rules can be found in [17].
- **Query classification** classifies the query according to query features that represent database primitives on the schema level (instances, properties, classes), on the operational level (e.g. aggregation and ordering), and on the structural level (conjuction, disjunction, property composition). The query features for the example question is shown in Figure 6.

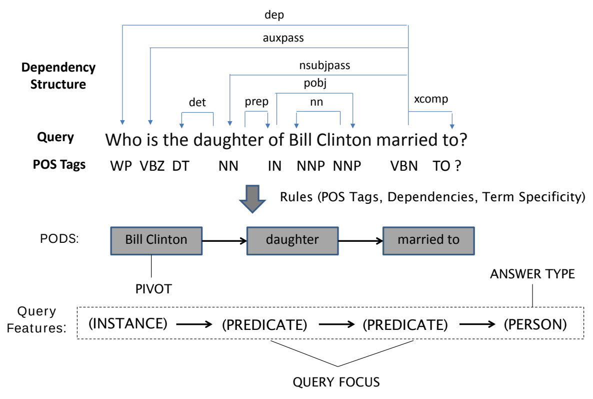


Fig. 6. Question analysis features for the example queries

After the query analysis, the PODS and the query features are sent to the *query planner*, which generates a query processing plan, involving the application of a sequence of search, navigation and transformation operations over the τ-Space:

- **Search operations** consist of keyword and distributional search operations over the data graph in the $\tau - Space$, in particular instance term search (over a term space), distributional class search, and property search.
- **Graph navigation and transformation operations** consist of graph composition and on the application of ordering, user feedback, aggregation and conditional operators.

The query planning algorithm, described in [15], orchestrates the search and graph navigation and transformation operations defined above. A query plan defines multiple operations over the index.

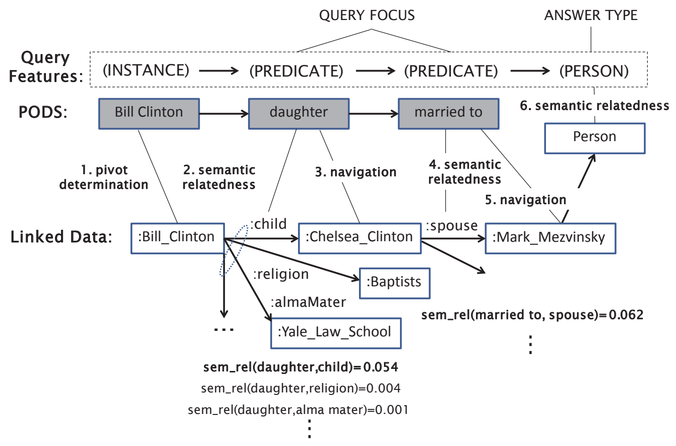


Fig. 7. Execution of a query processing plan for the query Who is the daughter of Bill Clinton married to?

With the PODS and the query features, the query processing approach starts by resolving the *core (pivot) entity* in the query (in this case Bill Clinton) to the corresponding database entity (`res:Bill_Clinton`), see Figure 7. The pivot determination depends on heuristics which take into account the query features and target the element which is less vague or ambiguous, and consequently presents a higher probability of a correct matching.

After Bill Clinton is resolved, the subspace of the entity `res:Bill_Clinton` is selected, constraining the search space to elements associated with this entity, and the next term in the PODS (daughter) is used as a query term for a distributional semantic search over the neighboring elements of `res:Bill_Clinton`. The distributional semantic search is equivalent to computing the distributional semantic relatedness between the query term (daughter) and all predicates associated with res:Bill_ Clinton (`dbo:religion`, `dbo:child`, `dbo:almaMater`, etc). The semantic equivalence between daughter and `dbo:child` is determined by using corpus-based distributional common-sense information, capturing that the words daughter and child occur in similar contexts. A threshold filters out unrelated relations. After daughter and `dbo:child` have been aligned, the query processing navigates to the entity associated with the property `dbo:child`, `res:Chelsea_Clinton`, and the next query term, i.e. married in our example, is considered. At this point the entity `res:Chelsea_Clinton` defines the search subspace (properties associated with `res:Chelsea_Clinton`) and a semantic search for predicates which are semantically related to married is condcuted. The query term married is matched to `dbo:spouse` and the answer to the query is found: the entity `dbpedia:Mark_Mezvinsky`. Figure 7 depicts the query processing steps for the example question. The high-level workflow and main components for the query approach are given in Figure 8.

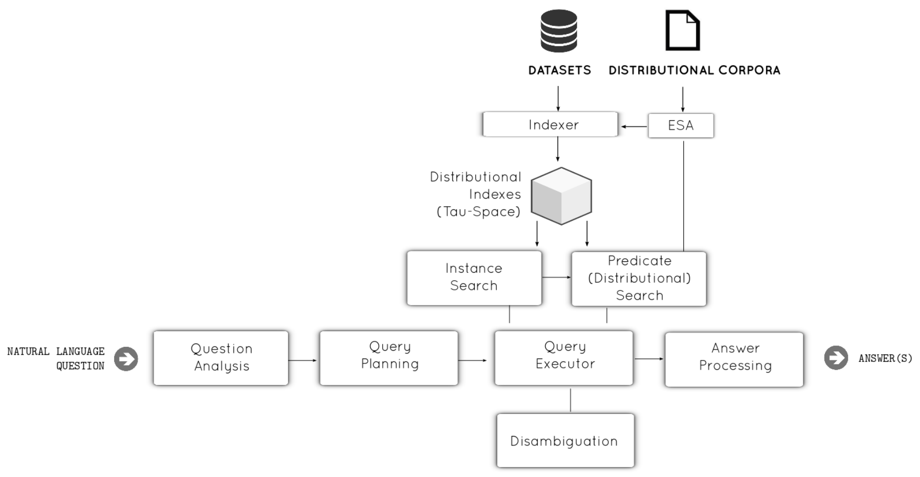


Fig. 8. High-level components diagram of the vocabulary-independent query approach and distributional inverted index structure

4.5 IBM Watson's DeepQA

The IBM Watson DeepQA system is a question answering system designed to be a machine contestant in the Jeopardy! quiz show, where three contestants compete against each another on answering open domain questions. The information sources for the DeepQA system are both unstructured and structured data. Despite the fact that the DeepQA approach focuses on extracting and scoring evidence from unstructured data, structured data sources (in particular in RDF format) play an important role. This section provides a high-level overview of the DeepQA system, focusing on the role of structured data in its question answering pipeline.

The DeepQA system is a massively parallel probabilistic evidence-based architecture which uses more than 100 different techniques for analyzing natural language, finding, merging scoring and ranking hypotheses. Below we describe the main components of the DeepQA platform, as depicted in Figure 9.

- **Question analysis:** consists of a mixture of question analysis methods including shallow and deep parsing, extraction of logical forms, semantic role labelling, coreference resolution, relations extraction, named entity recognition, among others.
- **Question decomposition:** consists in the decomposition of the question into separate phrases, which will generate constraints that need to be satisfied by evidence from the data.
- **Hypothesis generation** consists of two steps: *primary search* and *candidate answer generation*. The primary search step focuses on finding all content that can support the question (maximizing recall). Different information retrieval techniques for document and passage retrieval are used over

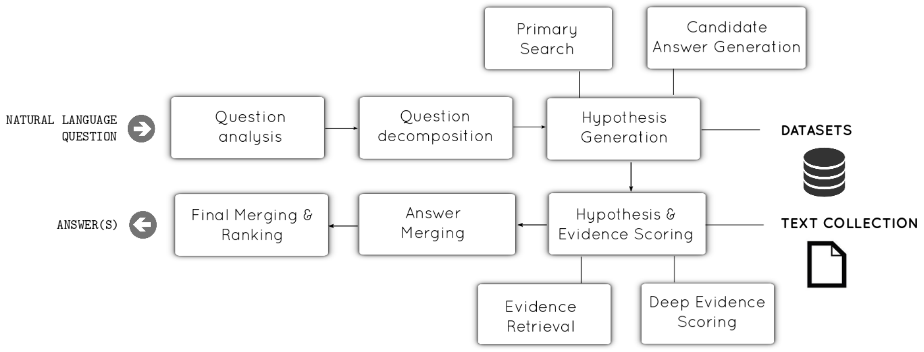


Fig. 9. Architecture of the IBM Watson DeepQA system

unstructured data sources and SPARQL queries are used over triple stores. Triple store queries in the primary search step are based on detected named entities, relations or lexical answer types in the question. The second step consists of the candidate answer generation, where information extraction techniques are applied to the search results to generate candidate answers (for example, for document search results from 'title-oriented' resources, the title is extracted as a candidate answer).

– **Soft filtering:** Consists in the application of lightweight (less resource intensive) scoring algorithms to a larger set of initial candidates to prune the list of candidates before the more intensive scoring components [13].
– **Hypothesis and evidence scoring:** consists of two steps: *supporting evidence retrieval* and *deep evidence scoring*. The supporting evidence retrieval step seeks additional evidence for each candidate answer from the data sources while the deep evidence scoring step determines the degree of certainty that the retrieved evidence supports the candidate answers. The system uses more than 50 scoring components that produce scores which range from probabilities and counts to categorical features. Scores are then combined into an overall evidence profile [13] which groups individual features into aggregate evidence dimensions.
– **Answer merging:** is a step that merges answer candidates (hypotheses) with different surface forms but with related content, combining their scores.
– **Ranking and confidence estimation:** is the last step, in which the system ranks the hypotheses and estimate their confidence based on the scores, using machine learning approaches over a training set. Multiple trained models cover different question types.

All of the components in DeepQA are implemented as Apache UIMA annotators. Apache UIMA[15] is a framework implementation of the *Unstructured Information Management Architecture*. UIMA also supports the parallelization of the pipeline components using asynchronous messages.

[15] http://uima.apache.org

Despite the fact that most of the evidence analysis in the IBM Watson DeepQA system is based on unstructured data, different components of the system rely on structured and semi-structured data and certain question types are answered by structured data sources. In particular, the DeepQA system uses three types of structured data sources:

– structured data available online (entity and associated types, movie databases)
– specific structured data extracted from unstructured data
– curated data providing additional information to the system (e.g. question and answer types)

RDF datasets such as DBpedia, Freebase and YAGO are used in the system. DBpedia is used to support the integration between unstructured and structured data and to assist information extraction tasks such as entity disambiguation and relation detection. Data cleaning techniques are used to normalize geospatial and temporal data in DBpedia into a common format. Freebase is used for geospatial data and YAGO as an entity type system. Disjointness properties are manually assigned at the higher level types in the YAGO taxonomy.

The DeepQA system processes a large set of questions that depend on temporal or geospatial evidence. Structured data sources play a fundamental role to provide geospatial and temporal evidence to the system. For example, temporal references detected at the question analysis step can be used to target specific temporal evidence over structured data sources. Structured data is also used for taxonomic reasoning and type coercion [12], checking whether a particular candidate answer's type matches the lexical answer type of the question [13].

Finally, structured data is also used in a separate complementary pipeline, for questions which require a more narrow but more precise answer, providing an incremental improvement in accuracy on the very small subset of questions for which it is applicable [12].

4.6 Other approaches

Exploring user interaction techniques, FREyA [10], an extension of the QuestIO system [11], is a question answering system which employs feedback and clarification dialogs to resolve ambiguities and improve the domain lexicon with the help of users. User feedback is used to enrich the semantic matching process by allowing manual query-vocabulary mappings.

RTV [20] is a system that integrates lexical semantic modelling and statistical inferences within an architecture that exploits Hidden Markov Models to select ontological triples matching the input question. The natural language interpretation task is decomposed into three different stages. First, salient linguistic information from the question, such as predicates, their arguments and properties, are extracted from the question. Second, this salient information is located in the ontology through joint disambiguation of all candidates. In particular, for each query a Hidden Markov Model is produced whose Viterbi solution is the comprehensive joint disambiguation across the sentence elements. Finally, the final query is executed against the RDF dataset. A similar system along these lines is the one by Ngonga et al. [33].

5 Do-it-yourself: Resources and tools

The following datasets play an import role for question answering over linked data, as they provide large amounts of cross-domain knowlegde:

- DBpedia
  `http://dbpedia.org/`
- Freebase
  `http://www.freebase.com/`
- YAGO and YAGO2
  `http://www.mpi-inf.mpg.de/yago-naga/yago/`
- Wikipedia dumps
  `http://dumps.wikimedia.org/`

Prominent tools for indexing and searching such datasets and text collections are the following ones:

- Lucene and Solr
  `http://lucene.apache.org/`
  `http://lucene.apache.org/solr/`
- Terrier
  `http://terrier.org/`

Building question answering systems is a complex task; it thus helps to exploit high-level tools for component integration as well as existing architectures for question answering systems:

- Apache UIMA
  `http://uima.apache.org`
- Open Advancement of Question Answering Systems (OAQA)
  `http://oaqa.github.io`
- Open Knowledgebase and Question Answering (OKBQA)
  `http://www.okbqa.org`
  `https://github.com/okbqa`

In the remainder of the section we provide a list of resources and tools that can be exploited especially for the linguistic analysis of a question and the matching of natural language expressions with vocabulary elements from a dataset.

Lexical resources

- WordNet
  `http://wordnet.princeton.edu/`
- Wiktionary
  `http://www.wiktionary.org/`
  API: `https://www.mediawiki.org/wiki/API:Main_page`
- BabelNet
  `http://babelnet.org/`

- FrameNet
  `https://framenet.icsi.berkeley.edu/fndrupal/`
- VerbNet
  `http://verbs.colorado.edu/~mpalmer/projects/verbnet.html`
- English lexicon for DBpedia 3.8 (in *lemon*[16] format)
  `http://lemon-model.net/lexica/dbpedia_en/`
- PATTY (collection of semantically-typed relational patterns)
  `http://www.mpi-inf.mpg.de/yago-naga/patty/`

Text processing tools

- GATE (General Architecture for Text Engineering)
  `http://gate.ac.uk/`
- NLTK (Natural Language Toolkit)
  `http://nltk.org/`
- Stanford NLP
  `http://www-nlp.stanford.edu/software/index.shtml`
- LingPipe
  `http://alias-i.com/lingpipe/index.html`

Dependency parsers

- MALT
  `http://www.maltparser.org/`
  Languages (pre-trained): English, French, Swedish
- Stanford parser
  `http://nlp.stanford.edu/software/lex-parser.shtml`
  Languages: English, German, Chinese, and others
- CHAOS
  `http://art.uniroma2.it/external/chaosproject/`
  Languages: English, Italian

Named Entity Recognition

- NERD (Named Entity Recognition and Disambiguation)
  `http://nerd.eurecom.fr/`
- Stanford Named Entity Recognizer
  `http://nlp.stanford.edu/software/CRF-NER.shtml`
- FOX (Federated Knowledge Extraction Framework)
  `http://fox.aksw.org`
- DBpedia Spotlight
  `http://spotlight.dbpedia.org`

[16] `http://lemon-model.net`

String similarity and semantic relatedness

- Wikipedia Miner
  http://wikipedia-miner.cms.waikato.ac.nz/
- WS4J (Java API for several semantic relatedness algorithms)
  https://code.google.com/p/ws4j/
- SecondString (string matching)
  http://secondstring.sourceforge.net
- EasyESA (semantic relatedness)
  http://treo.deri.ie/easyESA

Textual Entailment

- DIRT
  Paraphrase Collection: http://aclweb.org/aclwiki/index.php?title=DIRT_Paraphrase_Collection
  Demo: http://demo.patrickpantel.com/demos/lexsem/paraphrase.htm
- PPDB (The Paraphrase Database)
  http://www.cis.upenn.edu/~ccb/ppdb/

6 Evaluation campaigns

The two most prominent criteria for assessing the quality of an answer that a question answer system provides are:

- *Correctness*, i.e. whether the answer is factually correct.
- *Completeness*, i.e. whether the answer is complete (especially in the context of list or definition questions).

The following evaluation campaigns provide benchmarks for evaluating and comparing question answering systems over linked data with respect to these two criteria.

QALD The *Question Answering over Linked Data* (QALD) challenge[17] [27] aims to bring together researchers and developers from different communities, including NLP, Semantic Web, human-computer interaction, and databases. The core task of QALD is multilingual question answering over linked data, targeting all question answering systems that mediate between a user, expressing his or her information need in natural language, and semantic data. Given an RDF dataset and a natural language question or set of keywords in one of several languages (in QALD-4: English, Spanish, German, Italian, French, Dutch, and Romanian), participating systems are required to return either the correct answers or a SPARQL query that retrieves these answers.

[17] http://www.sc.cit-ec.uni-bielefeld.de/qald/

For the first instantiations of the challenge, the answers were to be retrieved from a single RDF dataset, in particular DBpedia and (although attracting a bit less interest) MusicBrainz. The fourth installment of the challenge, QALD-4, started to extend the task also to multiple, interlinked datasets, including questions that can only be answered by aggregating information from different biomedical RDF datasets, as well as hybrid question answering, including questions that require information from both structured RDF data and free text in order to be answered.

QALD evaluates systems with respect to a manually constructed gold standard that specifies SPARQL queries for each question. The answers a participating system provides are compared to the answers that the gold standard query retrieves. For each question q, precision, recall and F-measure are computed as follows:

$$\begin{aligned} \mathit{Recall}(q) &= \frac{\text{number of correct system answers for } q}{\text{number of gold standard answers for } q} \\ \mathit{Precision}(q) &= \frac{\text{number of correct system answers for } q}{\text{number of system answers for } q} \end{aligned}$$

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

On the basis of these measures, overall precision and recall values as well as an overall F-measure value is computed as the average mean of the precision, recall and F-measure values for all questions.

INEX Linked Data The goal of the Linked Data track of INEX[18] is to investigate retrieval techniques over a combination of textual and structured data, aiming to close the gap between key word search exploited in information retrieval and the reasoning techniques available for Semantic Web data. INEX Linked Data thus focuses on typical information retrieval questions, where structured data can be exploited to improve retrieval performance, e.g. because RDF properties provide additional information about semantic relations among objects that cannot be captured by keywords alone.

BioASQ The BioASQ project[19] organizes a challenge on biomedical semantic indexing and question answering. This includes a variety of tasks from different areas, such as hierarchical text classification, machine learning, information retrieval, question answering from text as well as structured data, and multi-document summarization.

Although the scope is much wider than just question answering over linked data, a lot of the assessed functionalities play an either directly or indirect role

[18] `https://inex.mmci.uni-saarland.de/tracks/lod/`
[19] `http://www.bioasq.org/`

for question answering, such as large-scale classification of biomedical documents and questions with respect to relevant ontology concepts, retrieval of relevant document snippets, concepts and knowledge base triples, as well as delivery of the retrieved information in a concise and user-understandable form.

Joint question answering lab at CLEF 2014 The above mentioned challenges – QALD, BioASQ and INEX Linked Data – recently joined forces in a joint question answering lab[20] at CLEF 2014. They all start from an information need expressed in natural language and build on the insight that the data sources required to answer that question could differ. Some questions might need to query structured data, especially if aggregations or logical inferences are required, whereas other questions might need querying free text and drawing textual inferences, and some questions may need both. By joining challenges on these different aspects, they aim to foster the general vision that question answering systems can find, process and integrate information from all available sources, no matter how diverse.

7 Trends and open topics

We showed the challenges involved in question answering over linked data as well as how state-of-the-art approaches address them. Despite the significant advances achieved in recent years, there is still a wide range of open topics that require future research.

Querying distributed linked data In the context of the Web, it can not be assumed that all relevant data is available on one site, so that answers to questions have to be found in scenarios where the data is distributed. While there are first approaches to this, they still assume that data is integrated locally and there is a central index that can be used to query data (e.g. Poweraqua). First approaches to federated and distributed querying exist (Treo, Ngonga et al), but the question of how to scale QA to the distributed Web of (linked) data is unanswered. A further challenge is related to yielding near real-time query processing, which state-of-the-art systems are still quite far away from.

Integration of structured and unstructured data A lot of information is still available only in textual form, both on the web and in the form of labels and abstracts in linked data sources. Therefore approaches are needed that can not only deal with the specific character of structured data but also with extracting information from several sources, processing both structured and unstructured information and combining such gathered information into one answer. Systems such as IBM's Watson have largely benefited from the integration of diverse data sources, including both unstructured and structured data [13,14].

[20] `http://nlp.uned.es/clef-qa/`

The integration of structured and unstructured data can benefit from both structured and unstructured resources and approaches. For example, named entities from linked RDF datasets can be used to support named entity recognition approaches over text sources by providing a first level structure for the domain, serving as a kind of indexing mechanism for unstructured resources in the context of question answering. From the opposite perspective, the information scale of unstructured data can make question answering systems over linked data much more useful in the short term and in industrial settings. Additionally, using structured resources to support more sophisticated information extraction approaches (such relation extraction) in the context of question answering can support the enrichment of structured datasets from unstructured data.

User interaction and context mechanisms Some of the question answering tasks can benefit from an interaction between the user and the system. For example, when interpreting the user's information need with respect to a particular dataset, user feedback can be used to remove ambiguity and vagueness, or to support co-reference resolution. In addition, allowing for a question answering dialog instead of single questions and answers would allow users to pose questions ina dialog context, e.g. refering to previous questions or answer, as in the following example:

– *User:* Who killed Martin Luther King?
– *System:* James Earl Ray.
– *User:* Was he captured?
– *System:* Yes.

Furthermore, the investigation of different interaction modalities may improve the efficiency of the question answering process, in particular in a dialog context. For example, speech recognition and synthesis may complement the text input, pointing gestures may provide specific references (see e.g. [4]) and cues, and data visualization techniques can provide data consumers with a more efficient interpretation of the answer.

Incorporating reasoning Despite the support of Semantic Web standards for deductive reasoning by grounding RDF graph data in Description Logics, logical reasoning is still very dependent on logically consistent knowledge bases, which is an unfeasible constraint for most datasets in the linked data cloud. However, allowing reasoning over inconsistent knowledge bases can strongly improve question answering over linked data. Additionally, investigating the role and the impact of different reasoning approaches (deductive, inductive, abductive, counterfactual, among others) under the question answering task is a long term and high impact research direction.

Moreover, the integration of common-sense knowledge external to the target datasets can be beneficial for addressing vocabulary variations and for supporting the reasoning behind the answer processing. The use of structured and unstructured common-sense resources such as Cyc[21], ConceptNet[22] and Wikipedia[23] in question answering is a fundamental component to improve the semantic flexibility of question answering systems.

Measuring confidence and answer uncertainty The complexity of the question answering task is associated with the variability and openness of the questions and the data. Providing confidence or uncertainty measures at each step of the question answering processing allows for an estimation of the quality of the final answer and can be used as a heuristic for the selection of different query processing strategies.

Multilinguality Current question answering systems over linked data are typically monolingual and there are so far no systems that are able to answer questions in more than one language. Extending QA systems over linked data to work for multiple languages involves bridging a severe vocabulary gap, as adding additional languages other than English exacerbates the lexical gap. In order to process questions in multiple languages, lexical knowledge about how a certain data property or class is expressed in the relevant languages, capturing all the intra-lingual and inter-lingual variance in how a vocabulary element can be expressed. A further bottleneck is that most QALD system require some preprocessing that is language-specific. Aqualog for example relies on JAPE-grammars that work for English only and extending the system to other languages requires the availability of grammars in multiple languages. Similar comments apply to Pythia and ORAKEL, as they rely both on language-specific grammars. The templates used by TBSL is also to some extend language-specific. Overall, all the systems discussed here are thus in general language-specific and substantial effort would be required to extend them to other languages. Treo would require semantic relatedness measures in different languages. Overall, developing systems that can answer questions in different languages is indeed a challenge which currently remains at the research frontier and has not been yet tackled.

Machine learning A new field dubbed *semantic parsing* has emerged recently that applies machine learning techniques to learn the mapping from natural language to semantics from training data. It is thinkable that such approaches are also applicable to the task of learning to map natural language questions to a SPARQL query, but so far Semantic Parsing approaches have not been thoroughly investigated in this setting. This remains for future work.

[21] `http://www.cyc.com/platform/opencyc`
[22] `http://conceptnet5.media.mit.edu/`
[23] `http://www.wikipedia.org/`

8 Conclusion

In this tutorial overview paper, we have introduced the problem of question answering over linked data. We have further presented the main challenges involved in the task and presented the typical architecture or anatomy of systems addressing the task. We have further provided an overview of state-of-the-art approaches to the problem, highlighting their features and drawbacks. Finally, we have described a selected set of systems, comprising those developed by ourselves, in more detail and concluded with a summary of open issues that need to be addressed in future research.

References

1. L. Androutsopoulos. Natural language interfaces to databases – an introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995.
2. Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.
3. Abraham Bernstein, Esther Kaufmann, and Christian Kaiser. Querying the semantic web with GiNSENG: A guided input natural language search engine. In *15th Workshop on Information Technologies and Systems, Las Vegas, NV*, pages 112–126, 2005.
4. Björn Bringert, Robin Cooper, Peter Ljunglöf, and Aarne Ranta. Multimodal dialogue system grammars. In *Proceedings of DIALOR'05, Ninth Workshop on the Semantics and Pragmatics of Dialogue*, pages 53–60, 2005.
5. Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. QAKiS: an open domain QA system based on relational patterns. In *Proceedings of the ISWC 2012 Posters & Demonstrations Track*, volume 914 of *CEUR Workshop Proceedings*, 2012.
6. Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.
7. Philipp Cimiano. Flexible semantic composition with DUDES. In *Proceedings of the 8th International Conference on Computational Semantics (IWCS'09)*, 2009.
8. Philipp Cimiano, Paul Buitelaar, John McCrae, and Michael Sintek. LexInfo: A declarative model for the lexicon-ontology interface. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):29–51, 2011.
9. Philipp Cimiano, Peter Haase, Jörg Heizmann, Matthias Mantel, and Rudi Studer. Towards portable natural language interfaces to knowledge bases – the case of the ORAKEL system. *Data & Knowledge Engineering*, 65(2):325–354, 2008.
10. Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. FREyA: an interactive way of querying linked data using natural language. In *Proceedings of the European Semantic Web Conference Workshops*, volume 7117 of *Lecture Notes in Computer Science*, pages 125–138. Springer, 2012.
11. Danica Damljanovic, Valentin Tablan, and Kalina Bontcheva. A text-based query interface to OWL ontologies. In *6th Language Resources and Evaluation Conference (LREC'08)*, 2008.
12. A. Kalyanpur et al. Structured data and inference in DeepQA. *IBM Journal of Research & Development*, 56(3/4), 2012.

13. David Ferrucci et al. Building Watson: an overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.
14. Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, 2014. to appear.
15. Andre Freitas and Edward Curry. Natural Language Queries over Heterogeneous Linked Data Graphs: A Distributional-Compositional Semantics Approach. In *18th International Conference on Intelligent User Interfaces (IUI'14)*, 2014.
16. André Freitas, Edward Curry, and Sean O'Riain. A Distributional Approach for Terminology-Level Semantic Search on the Linked Data Web. In *27th ACM Symposium On Applied Computing (SAC 2012)*. ACM Press, 2012.
17. André Freitas, João Gabriel Oliveira, Seán O'Riain, Edward Curry, and João Carlos Pereira da Silva. Querying linked data using semantic relatedness: a vocabulary independent approach. In *Natural Language Processing and Information Systems*, pages 40–51. Springer, 2011.
18. Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*, pages 1606–1611, 2007.
19. Daniel Gerber and Axel-Cyrille Ngonga Ngomo. Bootstrapping the linked data web. In *Proceedings of the 10th International Semantic Web Conference (ISWC)*, 2011.
20. Cristina Giannone, Valentina Bellomaria, and Roberto Basili. A HMM-based approach to question answering against linked data. In *Proceedings of the Question Answering over Linked Data lab (QALD-3) at CLEF2013. Lecture Notes in Computer Science (to appear)*. Springer, 2013.
21. Asunción Gómez-Pérez, Daniel Vila-Suero, Elena Montiel-Ponsoda, Jorge Gracia, and Guadalupe Aguado de Cea. Guidelines for multilingual linked data. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics (WIMS'13)*, pages 3:1–3:12. ACM, 2013.
22. E. Kaufmann, A. Bernstein, and R. Zumstein. A natural language interface to query ontologies based on clarification dialogs. In *Proceedings of the 5th International Semantic Web Conference (ISWC)*, 2006.
23. Jin-Dong Kim and Kevin Bretonnel Cohen. Natural language query processing for SPARQL generation: A prototype system for SNOMED-CT. In *Proceedings of BioLINK SIG*, 2013.
24. Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1545–1556, 2013.
25. Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 556–562, 2002.
26. Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. PowerAqua: supporting users in querying and exploring the Semantic Web. *Semantic Web*, 3(3):249–265, 2012.
27. Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluation question answering over linked data. *Journal of Web Semantics*, in press.
28. Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Journal of Web Semantics*, 5(2):72–105, 2007.

29. Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. Is question answering fit for the semantic web? a survey. *Semantic Web Journal*, 2:125–155, 2011.
30. J. McCrae, G. Aguado de Cea, P. Buitelaar, P. Cimiano, T. Declerck, A. Gomez-Perez, J. Garcia, L. Hollink, E. Montiel-Ponsoda, and D. Spohr. Interchanging lexical resources on the Semantic Web. *Language Resources and Evaluation*, 46(4):701–719, 2012.
31. Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04. Association for Computational Linguistics, 2004.
32. Uma Sawant and Soumen Chakrabarti. Learning joint query interpretation and response ranking. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, pages 1099–1110, 2013.
33. Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Question answering on interlinked data. In *Proceedings of the 22nd International World Wide Web Conference (WWW)*, pages 1145–1156, 2013.
34. Thanh Tran, Haofen Wang, Sebastian Rudolph, and Philipp Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 405–416, 2009.
35. C. Unger, F. Hieber, and P. Cimiano. Generating LTAG grammars from a lexicon-ontology interface. In Srinivas Bangalore, Robert Frank, and Maribel Romero, editors, *10th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, 2010.
36. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM, 2012.
37. Christina Unger and Philipp Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *Natural Language Processing and Information Systems: 16th International Conference on Applications of Natural Language to Information Systems, NLDB 2011, Alicante, Spain, June 28-30, 2011, Proceedings*, pages 153–160. Springer, 2011.
38. Sebastian Walter, Christina Unger, and Philipp Cimiano. A corpus-based approach for the induction of ontology lexica. In *Proceedings of the 18th International Conference on Application of Natural Language to Information Systems (NLDB 2013)*, 2013.
39. Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 1107–1116. ACM, 2013.