Semantic Web 0 (2020) 1–0
IOS Press

Bio-SODA - A Question Answering System for Domain Knowledge Graphs

Ana Claudia Sima [a,b,c], Tarcisio Mendes de Farias[b,c], Maria Anisimova[a,c], Christophe Dessimoz[b,c,d], Marc Robinson-Rechavi[b,c], Erich Zbinden[a,c], Kurt Stockinger [a],

[a] *ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland*
[b] *University of Lausanne, Lausanne, Switzerland*
[c] *SIB Swiss Institute of Bioinformatics, Lausanne, Switzerland*
[d] *University College London, London, UK*

Abstract. The problem of question answering over structured data has become a growing research field, both within the relational database and the Semantic Web community, with significant efforts involved in question answering over knowledge graphs (KGQA). However, many of these approaches are specifically targeted at *open-domain* question answering, and often cannot be applied directly in complex *closed-domain* settings of scientific datasets.

In this paper, we focus on the specific challenges of question answering over *closed-domain knowledge graphs* and derive design goals for KGQA systems in this context. Moreover, we introduce our prototype implementation, Bio-SODA, a question answering system that does not require training data in the form of question-answer pairs for generating SPARQL queries over closed-domain KGs. Bio-SODA uses a generic graph-based approach for translating questions to a ranked list of candidate queries. Furthermore, we use a novel ranking algorithm that includes node centrality as a measure of relevance for candidate matches in relation to a user question. Our experiments with real-world datasets across several domains, including the latest official *closed-domain* Question Answering over Linked Data (QALD) challenge – the QALD4 biomedical task – show that Bio-SODA outperforms generic KGQA systems available for testing in a closed-domain setting by increasing the F1-score by at least 20% across all datasets tested. We also provide a new bioinformatics benchmark with complex queries drafted in collaboration with domain experts. The experimental results show that for these types of real-world queries, the advantage of Bio-SODA is even more significant by outperforming state-of-the-art systems up to 46% improvement in the F1-score.

Keywords: Question Answering, Knowledge Graphs, Ranking

1. Introduction

The problem of question answering over structured data has gained significant traction, both in the Semantic Web community – with a focus on answering natural language questions over RDF data stores [13, 53, 56] – and in the relational database community, where the goal is to answer questions by finding their semantically equivalent translations to SQL [30, 31, 43]. Significant research efforts have been invested in particular in *open-domain* question answering over knowledge graphs. These efforts often use the DBpedia and/or Wikidata knowledge bases that are composed of structured content from various Wikimedia projects such as Wikipedia. A growing ecosystem of tools is therefore becoming available for solving subtasks of the KGQA problem, such as entity linking [20, 35, 39, 44] or query generation [55]. However, most of these tools are specifically targeted at open-domain question answering, mostly over DBpedia [48], which cast doubts on their applicability to other contexts, such as for scientific datasets.

On the one hand, encouraged by the recent success of machine learning methods, several new benchmarks for training and evaluating KGQA systems have been published [17, 50]. On the other hand, not only are these datasets generally costly to construct for a *new* domain, since they involve manual curation, but also most of the existing ones are synthetic (*i.e.*, not based on real query logs). Moreover, most datasets are lim-

ited to DBpedia or Wikidata, which may not be representative of other, single-domain knowledge graphs - which we refer to as *closed-domain*.

For example, one of the major question answering datasets over DBpedia, LC-Quad [50], as well as its updated version, LC-Quad 2.0 [17], include only simple multi-fact questions that connect at most two facts. In other words, these queries cover at most two or three triple patterns, with a query graph of at most two hops. However, real-world questions may be much more complex. In particular, a study of SPARQL query logs [8] across multiple knowledge graphs, including DBpedia, has shown that a significant fraction of real-world queries have 10 triple patterns or more. It therefore remains unclear whether existing training sets can serve as representative for real-world question answering systems over knowledge graphs in general.

All in all, an important unknown still remains as to how many of the lessons learned in the open-domain question answering world can be easily applied to closed-domain datasets, such as scientific datasets. In these domains, an equivalent ecosystem of tools is not readily available. As a consequence, data access and retrieval remain challenging for domain experts who are not familiar with structured query languages. Lastly, the problem is further aggravated by the *scarcity of open-source* question answering systems that can also be tested outside of the DBpedia ecosystem.

To illustrate the general problem of question answering over knowledge graphs, consider the simple data model in Figure 1. Although only a toy data model, this helps illustrate problems stemming from ambiguity: a natural language question such as *"which drugs are used for asthma?"* cannot be easily translated without relying on external knowledge (*e.g.* training data), given that *used for* cannot be directly mapped to either of the two properties (*possibleDiseaseTarget* or *sideEffect*) shown in the figure. However, node centrality metrics, such as the PageRank score of nodes in the knowledge graph, can help capture "common sense" knowledge, for example that *asthma* is more commonly a *Disease*, rather than a *Side Effect*.

As a step towards bridging the current gap in closed-domain question answering, we introduce Bio-SODA, a system designed to answer questions across multiple domain knowledge graphs where no prior training data are available. Bio-SODA relies on a generic graph-based approach in order to translate natural language questions into SPARQL queries. Furthermore,

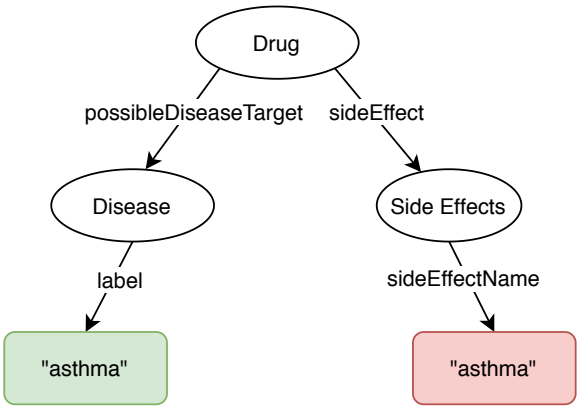

Fig. 1. Toy data model, simplified from the QALD4 benchmark datasets. Consider the following relatively straightforward question: *"which drugs are used for asthma?"*. In the QALD4 dataset, "asthma" appears as both a disease instance (shown in green), as well as a side effect (shown in red). The second interpretation describes drugs that can *trigger* asthma symptoms. Therefore, it is the opposite of the user's intended question. However, the predicate *used for* in the question cannot be easily linked to either of the properties indicated through arrows in the image. Due to ambiguity, the question is difficult to translate correctly in the absence of external knowledge, without relying on training data (*i.e.* to infer that *used for* implies a drug targeting a disease). In fact, as we will explain in a further example (see Figure 2), the correct interpretation is also on a *longer* path than the incorrect one. Traditional approaches that use the Steiner Tree algorithm (based on pairwise shortest paths computation) would therefore miss the correct interpretation. However, by taking into account node centrality metrics, the *Disease* asthma can be correctly ranked higher than the homonym *Side Effect* (for information about drugs triggering asthma symptoms as a side-effect, see the up-to-date Sider database entry at http://sideeffects.embl.de/se/C0004096/).

Bio-SODA is designed to compensate for incompleteness in the data—either due to missing schema information or, to some extent, due to missing labels. Although these situations should not occur when following ontology engineering best practices for representing data in RDF, our experience in working with real-world datasets shows that these problems are frequent in practice.

We make our prototype implementation available open-source[1]. The prototype enables both keyword search, as well as full question answering in English. We chose bioinformatics as our primary target domain, motivated by the rapid growth of publicly available RDF data in this domain. Specifically, around 8% of the Linked Open Data Cloud originates from the Life Sciences [27]. For the purpose of evaluating our

[1] Code at https://github.com/anazhaw/Bio-SODA

system, we use several real-world datasets stemming from different domains. For example, we use the latest *closed-domain* question answering challenge released as part of the official Question Answering on Large Databases (QALD) series, namely the QALD4 biomedical task [52]. Subsequent QALD challenges have since focused on open-domain question answering. We additionally note that domain-specific challenges, such as BioASQ [51], have focused less on question answering over *structured* data and more on information retrieval from unstructured text (documents), making the KGQA component only a small part of the required system to compete in the challenge.

Furthermore, we introduce a new benchmark of 30 representative question-answer pairs drafted in collaboration with domain experts from two different subfields of bioinformatics: orthology (two genes are called orthologs if they were the same in the last common ancestor) and gene expression (a measure of gene activity). The questions represent information needs related to two in-use and publicly available bio-databases, Orthologous MAtrix (OMA), a database of evolutionary relationships among genes found in different species [3], and Bgee, a gene expression database [5]. It has been shown that the number of triple patterns in the expected SPARQL answer queries is inversely proportional to the performance of question answering systems in generating these queries [45]. With an average complexity of 7 triple patterns per query, our catalog highlights that real-world questions in these domains surpass the current benchmarks in complexity and therefore require new approaches. Finally, as an application of Bio-SODA to an entirely different, non-bioinformatics domain, we use the CORDIS dataset describing European Union (EU) funded research projects[2].

This paper makes the following **contributions**:

- We introduce Bio-SODA—a novel question answering system over domain knowledge graphs that *does not require prior training data* (question-answer pairs) for translating natural language questions into SPARQL.
- We define a novel ranking algorithm for selecting the best automatically generated SPARQL statements in response to a given natural language question. The ranking algorithm combines *syntactic and semantic similarity*, as well as *node centrality* in the knowledge graph. Many existing question answering systems either rely on simple metrics for ranking, such as the length of the answer query graph [43], or require extensive training data in order to learn a ranking function [33]. To the best of our knowledge, our approach is the first to take into account all three factors (syntactic and semantic similarity, as well as node centrality) for ranking candidate queries.
- Our experiments on various real-world datasets show that Bio-SODA outperforms state-of-the-art KGQA systems by 20% on the F1-score using the official QALD4 biomedical benchmark and by an even higher factor on the more complex bioinformatics dataset. These differences generally stem from the higher complexity of the questions composing these datasets compared to existing benchmarks in open-domain question answering: higher number of triple patterns per query, but also extensive use of literals (numbers, strings), filters, negations, and so on. We make the bioinformatics question answering benchmark targeting Bgee and OMA publicly available[3] for future research in this direction.
- Finally, we outline the challenges of *closed-domain* question answering and identify requirements for generic question answering systems over domain knowledge graphs.

The rest of this paper is structured as follows: we introduce some of the specific challenges of closed-domain question answering in Section 2. We illustrate the question answering pipeline, through a concrete example from the biomedical domain, in Section 3. We present the system architecture of Bio-SODA in Section 4. Next, we describe the datasets used for evaluation, their specific challenges and the results obtained in Section 5. In Section 6 we discuss lessons learned from building a question answering system for real-world domain datasets. Section 7 places our contribution in the context of the related work. We conclude by outlining directions for future work in Section 8.

2. Problem Statement

In this section we introduce some of the specific challenges of closed-domain question answering, which shape the architecture of the Bio-SODA system (described in Section 4).

[2] https://cordis.europa.eu/projects

[3] See Benchmarks folder in the Bio-SODA github repository

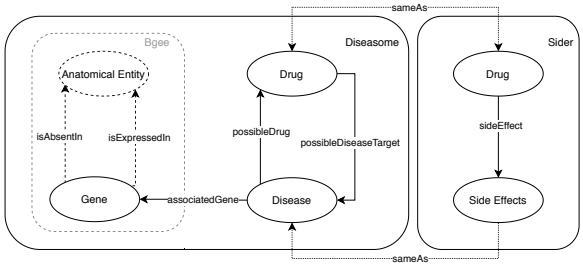

Fig. 2. Simplified data model, inspired from the QALD4 Diseasome, Drugbank and Sider datasets, as well as the Bgee data model (which includes the *Anatomical Entity* class). The data model is a multigraph, including disjoint properties – such as *isAbsentIn* and *isExpressedIn*, as well as inverse properties, such as *possibleDiseaseTarget* and *possibleDrug*. The information represented by one of these properties can be derived from the another one. This is an example of redundancy at the data level that could be avoided. Furthermore, the *sameAs* property between instances of the two distinct *Drug* classes represents the join point between the two datasets. To make matters more complicated, a *Side Effect* and a *Disease* can be described by the same terms, with instances of the two classes being related via the *sameAs* property. As a result, even simple questions such as *"which drugs might lead to strokes?"* are hard to automatically translate correctly in the absence of external knowledge (*i.e.* "lead to" = "side effect" as opposed to "possible disease target").

– **Lack of training data.**
For many of the closed-domain knowledge graphs there is no sufficiently long and diverse log of queries in order to derive a representative training set for a machine learning-based solution. So far, existing training corpora have proven costly to construct [50], with the added drawback that any semi-automatically generated dataset risks compiling a set of question-answer pairs that are non-representative for the information needs of real users of the KGQA system, in this case, domain experts.
– **Specialized terminology and vocabulary.**
Closed-domain knowledge graphs, such as scientific datasets, often make use of specialized vocabularies.
On the one hand, this means that data-driven methods for entity and property linking (*e.g.* N-gram-based matching, using an inverted index over the data) are likely to perform better than "off-the-shelf" solutions. These solutions mostly revert to open-domain knowledge graphs for linking terms, such as Falcon [44] or Spotlight [35] for DBpedia. However, *specialized terms cannot always be linked* to DBpedia.
On the other hand, the target user is a domain expert and is therefore expected to be more familiar with the domain terminology. As a result, the lexical gap [23] and question ambiguity, which are common in open-domain question answering, may be smaller problems in the context of closed-domain question answering.
– **Rule-based approaches perform well, but are costly to build and maintain.**
So far, state-of-the-art solutions for closed-domain question answering have been mostly rule-based systems, relying on manually handcrafted rules. For example, GFMed [34] and Pomelo [25], the top 2 ranked systems in the QALD4 biomedical challenge, have achieved very good results in the challenge, but at the cost of very little generality. In essence, these systems suffer from significant overfitting: to be applicable to a new domain, their rule sets would need extensive or even complete rewriting. Moreover, even for a new dataset within the same domain, for which the schema differs, new rules need to be added in order to accommodate the differences.
In some cases it is beneficial to incorporate a minimal set of rules in KGQA systems, particularly for deriving complex concepts. However, this should be a last resort and not the main translation mechanism, given that a large rule set is hard to maintain and scale.
– **Schema-less, incomplete data with imprecise labels.**
Real-world datasets, particularly RDF knowledge graphs, often exhibit poor structure [28, 41], for example, properties with missing or generic domains and ranges. In such cases, a question answering system that needs an explicit data schema should first enrich the existing schema, for example, by inferring property ranges and domains based on instance-level data.
An added challenge is the *quality of the RDF labels* assigned to the data. Given the manual nature of the annotation process of most RDF data, it can be error-prone, leading to imprecise or missing labels. Note that many properties in the official QALD4 biomedical dataset do *not* have any labels associated. For example, <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/foodInteraction> does not have any label associated in the official challenge dataset, although 4 (out of 50) questions specifically target this property. In these cases, a question answering system should try to infer definitions based

on other available information, for example, Uniform Resource Identifier (URI) fragments.

Label *imprecision* can also affect the performance of question answering systems, given that it hampers correct entity linking. Consider the following example: let us assume an RDF dataset with two different classes: class A, with a label “Gene Tree" and class B, with a much more verbose label, such as “This class describes genes along with their identifiers according to the information in PubMed January 2020". A search for the term “gene tree" would indeed return a single candidate match, namely class A. However, if the user would instead search for “gene", both class A and class B are potential matches. Although according to pure string similarity measures class A is a better match, it is clear that in fact class B should be returned. This shows that pure string similarity metrics are not always sufficient for entity linking. For an extended study on the quality of labels in the web of data, see [19].

Finally, a related problem in scientific datasets and particularly prevalent in biological data, is the *complexity of ontologies*. These can define thousands of terms, out of which, however, only few are actually *used*. For example, the Gene Ontology [4], widely instantiated by bioinformatics databases, contains many terms not used at all, for example the term GO:0062165[4]. However, these unused terms cannot be removed since they are part of an external ontology that cannot be changed. This suggests that KGQA systems using these data should take into account additional metrics for entity linking, such as the importance of a term in the knowledge graph, in order to reduce the number of potential matches for a given user term. This can be achieved, for example, by taking into account the PageRank score of the matched node in the knowledge graph.

– **Integration with external datasets.**
Since a closed-domain knowledge graph might be an information silo, it is often beneficial to integrate it with external sources in order to obtain new knowledge. In fact, an increasing number of domain datasets are being made available in the form of knowledge graphs for this exact purpose [6, 18]. Figure 2 illustrates the integration of Diseasome and Sider (both part of the QALD4 biomedical task) through the *sameAs* equivalence relationship between the instances of two *Drug* classes. This enables joint queries across the datasets. A further integration could be done between Diseasome and Bgee, based on instances of the *Gene* class.

A question answering system for closed-domain knowledge graphs should therefore also be easily extensible to new datasets, as well as updates in the existing data sources. Ideally, these updates should not require the re-indexing of all previously available data for each new release. As a result, the system should support incremental indexing.

– **High level of redundancy.**
Although not specific to closed-domain knowledge graphs, redundancy can also be an issue in this context, even more so when multiple datasets are integrated (as is the case for the QALD4 benchmark considered in this article). Redundancy can occur both at the schema level, as well as at the instance level. For example, in Figure 2, the *Drug* concept is defined by two different classes, whose instances are however connected via *sameAs* predicates. The *Drug* concept is therefore the *join point* between the two integrated datasets, *Diseasome* and *Sider*. Moreover, the *Drug* and *Disease* concepts are connected through two different properties, *possibleDrug* and *possibleDiseaseTarget*, that essentially represent the same information. This results from the integration of *Diseasome* and *DrugBank* and in principle implies that one of the two properties can be inferred from another one (i.e. $(x, y) \in R \Leftrightarrow (y, x) \in R^{-1}$ where R^{-1} is the inverse relation of an R relation). Furthermore, redundancy can also occur at the instance level: in the EU projects dataset, the same organization name can be assigned to an instance of the *Organization* class, as well as to an instance of a *Participant* in a project.

[4] See https://www.ebi.ac.uk/QuickGO/term/GO:0062165

3. Question Answering Pipeline

In this section we introduce the question answering pipeline in Bio-SODA. For this purpose, we start from the following motivating example. Assume that a domain expert is interested in answering the question:

“*What are the drugs for diseases associated with the BRCA genes?*"

Note that, based on the biomedical literature, mutations in the two *BRCA* genes, BRCA1 and BRCA2 (stemming from *BReast CAncer*) are known to be associated with multiple types of cancer. Moreover, assume that the domain expert aims to answer the question using information from the QALD4 biomedical dataset, which integrates three different sources: *Drugbank*, a database containing information about drugs; *Sider*, a database describing the side-effects of drugs; and *Diseasome*, a database about diseases, as well as possible drugs that can be prescribed for each disease.

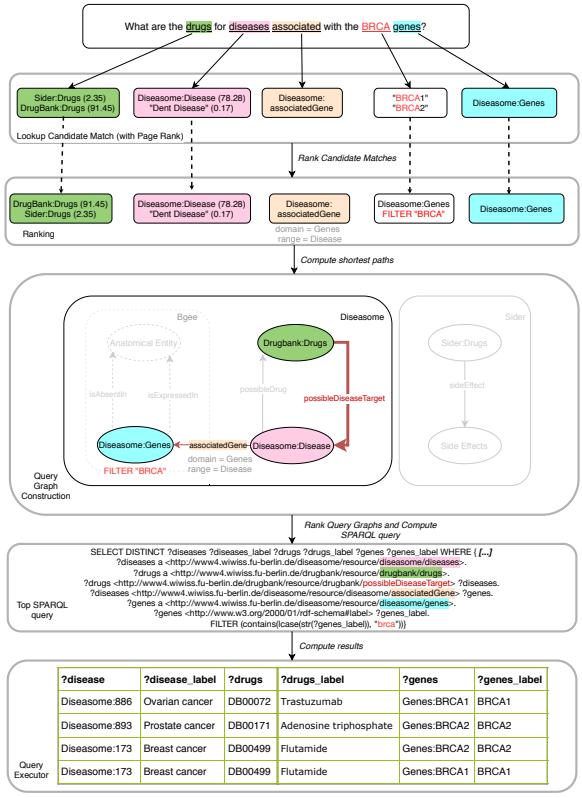

Fig. 3. Simplified answer pipeline for the query "*What are the drugs for diseases associated with the BRCA genes?*". For the sake of simplicity, PageRank scores are solely displayed when more than one match is found.

The question answering pipeline of Bio-SODA is illustrated in Figure 3. The main steps involved in the answering process are the following: first, the system retrieves, from an Inverted Index, matches for all tokens in the user question. In this example, all tokens are of length one, *i.e.* composed of a single word. The index will enable retrieving not only the URI of the candidate match, but also its PageRank score (an example is shown in parentheses for the first two tokens in the Figure), as well as the class and property names of the match (omitted in the figure for simplicity). For example, the lookup for "*BRCA*" retrieves instances of the class *Diseasome:Genes*, where the *rdfs:label* property matches the user token ("*BRCA1*", "*BRCA2*"). A few simplified Inverted Index entries are provided in Table 1.

In a second step, candidates are grouped together according to class and property (a `FILTER` for the token *BRCA* is created on the *Diseasome:Genes* class), as well as ranked according to string similarity and PageRank score. From here, the algorithm detailed in the following sections (see Algorithm 1) is applied to construct candidate query graphs. For simplicity, Figure 3 only shows the query graph obtained for the top ranked candidate matches. However, Bio-SODA will generate multiple alternative interpretations, for example, also including the interpretation considering *Sider:Drugs* instead of the *DrugBank:Drugs*. This can be tested in the demo page of Bio-SODA for QALD4. From the query graph, the corresponding SPARQL query is generated by also including labels for diseases and drugs. Finally, the result set shown in the bottom of Figure 3 is returned.

In the following section, we introduce the system architecture of Bio-SODA and describe the design of the question answering pipeline in more detail.

4. System Architecture

In this section, we describe the Bio-SODA system architecture, shown in Figure 4. The main building blocks of the system are the following:

- *Preprocessing Phase*: This phase includes building indexes for efficient lookup as well as generating a summary graph, i.e. schema graph, which will serve as the basis for constructing candidate SPARQL queries in response to user questions. This phase is only executed once, when initialising the system.
- *SPARQL Query Generation Phase*: This phase represents the question answering process and includes (1) looking up query tokens in the database, (2) ranking the candidate tokens, (3) constructing the candidate query graphs, (4) ranking the query graphs in order of relevance to the user question; and finally (5) constructing a valid SPARQL query and presenting the results.

We will now discuss these phases in more detail.

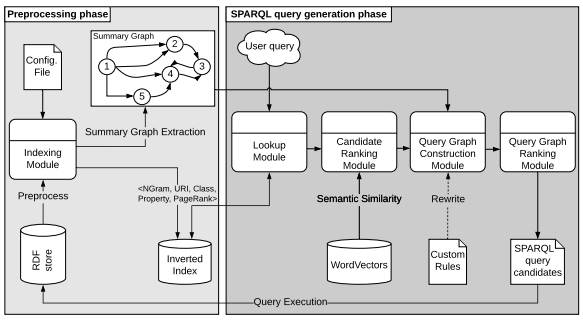

Fig. 4. Bio-SODA System Architecture

4.1. Preprocessing Phase

The core component of this phase is the *Indexing Module*, which extracts the *Inverted Index* as well as the *Summary Graph* of the RDF data sources:

- *Inverted Index*: stores the vocabulary of the system. More precisely, all the properties that should be searchable from the RDF data store are indexed, according to a configuration file that specifies the list of properties of interest (by default, all string literals will be indexed). A further configuration option is whether URI fragments should also be parsed and indexed. In this case, these fragments are split by a predefined punctuation list, and through a camel case regex (*e.g.*, "possibleDiseaseTarget" will be indexed as the corresponding keywords "possible disease target").
  The inverted index is stored in a relational database for fast searches and it is used to match tokens (sequences of keywords in a user query) against the RDF data. More precisely, the index stores: keywords (N-grams of literals indexed), the indexed instance URI, the class of this instance, the property from which the keywords were indexed (*e.g.* label), as well as the PageRank score of the instance. PageRank scores are computed using the approach presented in [16]. We further explain the role of each indexed information item in Subsection 3. An example is shown in Table 1.

- *Summary Graph Extractor*: used in order to enrich the schema of the knowledge graph(s) using instance-level data from the RDF store. The Summary Graph is essentially an accurate *schema of the integrated RDF data*[5]. Moreover, the Summary Graph serves as the basis for constructing candidate query graphs from selected entry points (*i.e.*, matches for tokens in a user question).
  Computing a Summary Graph allows the system to compensate for incomplete schema information, for example, in cases where domains and ranges for properties are either missing or ill-defined. A second benefit of the Summary Graph is that it enables integrating multiple schemas from different knowledge graphs.
  Extracting the summary graph is achieved via SPARQL queries that compute, for example, domains and ranges of all properties, based on the classes of the instances which they connect. As a simplified example, a triple asserting "Migraine ->possibleDrug ->Ibuprofen" will result in *Disease ->possibleDrug ->Drug* being added to the Summary Graph.
  We currently assume as a minimum requirement that each instance in the RDF data has a well-defined class, *i.e.* an explicit *rdf:type*. If this is not the case, additional preprocessing with external tools (for example, using RDF schema discovery techniques [28]) is required in order to properly define types for all RDF instances.

We note here that indexing is a pre-processing step that is only required once, when the system is initialized. Afterwards, updates to the RDF store can be incorporated periodically through incremental updates (appends) to the inverted index, while the Summary Graph will only need to be recomputed in case of schema changes.

4.2. SPARQL Query Generation Phase

Given a natural language question, the goal of the Bio-SODA system is to translate it into a set of ranked candidate SPARQL queries, such that the top ranked query is the closest to the user's query intent. In the following, we detail the role of each component involved in this translation process, namely the *Lookup Module*, the *Candidate Ranking Module*, the *Query Graph Construction Module*, the *Query Graph Ranking Module* and the *Query Executor Module*.

[5]Note that multiple RDF sources can be combined, as long as they are semantically aligned - *i.e.* they have at least one common concept, such as *Gene*.

Lookup Key	URI	Class	Property	PageRank
stroke	side_effects:C0038454	sider:side_effects	sider:side-EffectName	0.34
drug	drugbank:drugs	owl:Class	rdfs:label	91
drug	sider:drugs	owl:Class	rdfs:label	2.3
possible disease target	diseasome:possible-DiseaseTarget	rdf:Property	uri_match	80

Table 1

Inverted Index Sample. The lookup key is used for fast searches based on keywords from a user question. The remaining information is used in attaching candidate matches to the Summary Graph (see description in Section 4) in order to construct the corresponding query graphs. A lookup key can consist of multiple keywords. The same lookup key can appear multiple times.

– *Lookup Module*:
The lookup module has the role of retrieving the best candidate matches for tokens identified in a user query. A token is defined by the longest sequence of keywords that matches an entry in the inverted index. For example, in the question "*What are the possible disease targets of Ibuprofen?*" the two tokens extracted will be "*possible disease target*" (corresponding to an RDF property name) and "*Ibuprofen*" (corresponding to one or more Drug instances).

– *Candidate Ranking Module*:
The lookup module can return a large number of candidate matches per token. Due to this, the ranking module helps *group together equivalent matches and rank them* in order of relevance to the initial query. More precisely, instances of the same class and with an identical property that assigns the matched values are grouped together into a single candidate. For example, instances of the class *Drug* with matching *rdfs:label* are grouped together. A concrete case would be the genes *BRCA1* and *BRCA2* as a match for the keyword *BRCA* in a natural language question (see question illustrated in Figure 3). The single candidate, in turn, will be used to create a FILTER in the SPARQL query.

Furthermore, both *string similarity* and *node importance* are taken into account when ranking. Including the PageRank score as a measure of importance in the knowledge graph reduces the influence of the quality of labels assigned (labels which can be imprecise, see discussion in Section 2). The intuition behind this is that domain knowledge graphs usually cluster around a few important concepts, which will be reflected in the PageRank scores of the corresponding nodes. For example, UniProt[6] [42], a protein knowledge base containing more than 60 billion triples, includes only 177 classes at the time of writing. Out of these, only few classes, such as *Protein* and *Annotation*, have a central role, and will usually be the target of domain expert questions on this source. Likewise, in the case of the Cordis EU projects dataset, two different classes of Projects are available, *EC-Project* and *ERC-Project*. However, there is significantly more information in the dataset for the first class. In the lack of query logs or handcrafted rules for mapping query tokens to the correct candidates, the PageRank score can serve as a good proxy for ranking candidates according to node centrality, similarly to the initial approach used by web search engines [40].
As an added benefit, scoring with PageRank also ensures that metadata matches are prioritized. For example, *Drug* as a class name will rank higher than an instance match.
Finally, to ensure that candidate matches not only have good string similarity, but are also *semantically similar*, word embeddings are also used in the candidate ranking. The similarity comparison ensures that spurious matches, such as *gene* compared to *oogenesis*, are discarded based on a pre-defined similarity threshold in the system configuration. Any word embeddings can in principle be used with Bio-SODA. For the two main bioinformatics use cases considered in this paper, we use Word Vectors extracted from PubMed, as described in [36]. The candidate ranking module presents to the user top N matches per query token, where N is configurable in the system.

– *Query Graph Construction Module*:
The goal of this module is to use the matches from the previous step to *generate a list of candidate*

[6] https://sparql.uniprot.org/

Algorithm 1: Iterative graph-based approach for constructing query graphs from candidate matches

Data:
$M^{n\times t}$: the matrix of ranked candidate matches, where
n = the number of candidate matches per token,
t = the number of tokens in the user question.
M_i = a set of candidates covering one match per token (i.e. the i^{th} row vector of the $M^{n\times t}$ matrix).
G: Summary Graph of the RDF data
Result: S: the ranked set of candidate query graphs

```
foreach M_i ∈ M do
    QG_i = φ (empty graph)
    foreach candidate match T_j ∈ M_i do
        if T_j = a RDF property then
            Get domain D and range R of T_j from G;
            Add D and R as vertices to QG_i;
            Add edge T_j between D and R in QG_i;
            if multiple domains / ranges for T_j then Create a new copy of QG_i per alternative;
        else
            Compute in summary graph G: shortest paths between class of T_j and classes of other matches T_z in M_i;
            Add shortest paths to QG_i
            if multiple alternatives exist then
            Create a new copy of QG_i per alternative;
        end
    end
    Add spanning tree extracted from QG_i to result set S (Steiner tree approximation)
end
S_sorted = sort S by sum of match score of composing vertices. On a tie, sort by the weight (i.e. the number of edges) of spanning tree.
return S_sorted
```

query graphs. We extend the approach presented in [22] to translate matches to query graph patterns. More precisely, we apply the iterative algorithm shown in Algorithm 1: for each set of candidate matches (one match per query token), we augment the Summary Graph by attaching the candidate matches to their corresponding class. Next, we find the minimal subgraph that covers all matches. For this purpose, we solve the approximate Steiner tree problem by computing the minimal spanning tree that covers one match per token.

Note that there might be multiple such subgraphs, given that two classes can be connected via multiple properties. However, unless the user can be involved in disambiguating, it is important to generate all the variants, given that two equal-length subgraphs might actually have opposite semantics. Recall the example shown in Figure 2, where the properties *e.g*, *isAbsentIn versus isExpressedIn* both connect the same two classes, but represent disjoint result sets.

Finally, in some cases handcrafted rules for inferring new concepts or relations are required, due to the complexity of the corresponding query graphs. In such cases translating user questions into SPARQL cannot be done via simple entity linking methods. Therefore, if needed, our approach also supports adding rules to derive implicit information from the original knowledge graph as part of the question answering pipeline. These rules are implemented as sub-queries similar to the `SELECT` SPARQL query form. In this case, the rule head is the SPARQL query projection, and the rule body is the WHERE clause content.

An example for the evolutionary relationship "ortholog", describing genes that descend from the same common ancestor gene, is shown in Table 2. The problem of identifying the orthologs of a gene can be considered similar to finding persons descending from a common ancestor at a certain level in an ancestry tree. The *ortholog* relation is not explicitly available as an RDF property in the OMA RDF data, but rather asserted through a complex set of property paths[7]. An extended discussion on this is available in [10]. By providing custom rules, our approach enables semantically enriching the data sources and handling complex concepts which translate to long query subgraphs.

[7] https://www.w3.org/TR/sparql11-query/#propertypaths

Relation	Query Graph
?X ortholog ?Z	prefix orth: <http://purl.org/net/orth#> ?X a orth:Protein. ?Z a orth:Protein. ?cluster a orth:OrthologsCluster. ?cluster orth:hasHomologousMember ?node1. ?cluster orth:hasHomologousMember ?node2. ?node2 orth:hasHomologousMember* ?X. ?node1 orth:hasHomologousMember* ?Z. filter(?node1 != ?node2)

Table 2
Example of a custom rule for orthology data

– *Query Graph Ranking Module*:
The query graph ranking module plays an important role in presenting the user with a meaningful, ordered list of results. In contrast to existing work, we do not return the *overall minimal subgraph* as the top result, but rather the *graph that maximizes the sum of the match scores* of the candidates covered. To understand why this is the case, consider the following question: *"what are the drugs for asthma?"*. This question translates to a 2-hop query graph, joining *Drug* and *Disease* via the *possibleDiseaseTarget* path (see Figure 2). However, one likely scenario is that the description of a *Drug* instance includes the keyword *asthma*. In this case, the minimal query graph would be 1-hop only, retrieving only *Drug* instances that explicitly contain the keyword in the description, probably a small subset of all instances which have the corresponding *Disease* as a possible target. In this case, the minimal result would have good precision, but very low recall.

– *Query Executor Module*:
Finally, the query executor translates the ranked query graphs into SPARQL queries, assigning meaningful variable names, also adding human-readable fields to the result set, as shown in Table 6. Importantly, we do not only return the best result, but rather a ranked list of possible interpretations (top N, where N is configurable in the system). This gives the user the opportunity to inspect the results in order to use further solely the interpretation (*i.e.* SPARQL query) that matched the question intent.

5. Evaluation

5.1. Datasets

In this section we introduce the 3 datasets considered for evaluating Bio-SODA. The characteristics of these datasets are shown in Table 3. Importantly, all three are real-world, in-use datasets—here we highlight particular challenges that need to be overcome for each dataset in the context of designing a generic question answering system:

1. The QALD4 biomedical dataset is composed of Sider, DrugBank and Diseasome. The particularity of this dataset is that it contains a lot of redundancy, such as multiple *Drug* classes, identical terms describing both *Disease* and *Side Effects* instances, which are connected via *owl:sameAs* properties, and so on.
2. The bioinformatics dataset is composed of the Bgee (gene expression) and OMA (orthology) RDF stores. Given the highly specialized domain information contained in these sources, a particularity of this dataset is that questions can include complex concepts which translate to long SPARQL query graphs. An added challenge deriving from this is that the same concepts can be connected through multiple equal-length paths with semantically different or even opposite meanings.
3. The CORDIS dataset of EU-funded projects. Although this dataset has a simpler schema, the challenge here is that questions can have a higher degree of ambiguity. In some cases, multiple interpretations are valid – for example, many terms are reused often and in a variety of contexts, such as "*Big Data*". This can be either part of a project title, a topic, an organization name, and so on. Therefore, identifying the query intent in some cases (*e.g. Show Big Data projects*) cannot be done without user disambiguation.

5.2. Queries

We have reused the official 50 queries of the QALD4 biomedical challenge[8]. We do not distinguish between training and test queries. Indeed, we report performance metrics for all systems we tested across the entire set of 50 queries. Given that the test set

[8]https://github.com/ag-sc/QALD/blob/master/4/data

Dataset	Sources	#Classes	#Triples	Size on Disk
QALD4-biomedical	Drugbank, Diseasome, Sider	12	0.69 M	200 MB
Bioinformatics	Bgee, OMA	37	430 M	30 GB
CORDIS	EU projects dataset	26	6.5 M	1 GB

Table 3
Descriptions of the 3 public datasets used in our evaluation.

was also available to participants in the official challenge, we believe this to be a fair evaluation. We do not change the questions in the official challenge, not even in cases where we could identify mistakes in the question. Furthermore, as opposed to previous work using this benchmark [49], we do not materialize triples based on *owl:sameAs* statements and only use the exact dataset, as provided in the official benchmark.

For the bioinformatics dataset, we created in collaboration with domain experts, a benchmark of 30 queries, in increasing order of complexity, across two datasets, namely Bgee and OMA. The queries represent real information needs of domains experts within the field of gene expression and orthology, using the publicly available RDF data of Bgee[9] and OMA[10]. The average number of triple patterns per query here is 7 (not taking into account joint queries between the two sources, which have even higher complexity), with some questions jointly targeting 4 entities or more (*Gene*, *Species*, *Anatomical Entity*, *Developmental Stage*). In contrast, in existing benchmarks, such as LC-Quad [50], queries with only 2 entities are already considered complex.

In order to test Bio-SODA using an entirely different domain, using the CORDIS dataset of EU funded projects, we created a test set of 30 queries in increasing order of complexity. Given the relatively simple structure of this data model, the average number of triple patterns per query is close to that of existing KGQA benchmarks [50], with an average 2.3 triple patterns per query. However, the complexity stems from the usage of filters, literals in the query, as well as the higher degree of ambiguity.

Queries across the three datasets include aggregations, negations, and make extensive use of filters.

All questions, as well corresponding SPARQL queries, are available in the Evaluation folder of our GitHub repository[11].

[9] https://bgee.org/sparql

[10] https://sparql.omabrowser.org/sparql

[11] Evaluation in https://github.com/anazhaw/Bio-SODA/

5.3. Results

In this section we compare the performance of Bio-SODA against state-of-the-art systems which are open-source. Although there is a rich literature introducing question answering systems for knowledge graphs in recent years (to cite a few examples, see [15, 46, 57] etc.), only a few of them are publicly available for testing. Some provide at most a web service interface, where the target knowledge base usually cannot be changed[12].

Given these constraints, we focused on open-source systems that are publicly available for testing, in particular systems that show state-of-the-art performance according to a recent survey on natural language interfaces to databases [2]. Specifically, we tested Sparklis [21], a generic query builder system for knowledge graphs[13]. Moreover, we compared against GFMed [34] which was top ranked in the QALD4 biomedical challenge and specifically designed for this dataset. We use GFMed's publicly available grammar[14] to evaluate how the system performs outside of the official QALD4 biomedical dataset. In addition, we compared our approach against SQG [55], a system for query generation over knowledge graphs[15].

We use the standard evaluation metrics of precision (P), recall (R) and F1-score, macro-averaged over all questions in the dataset. For Bio-SODA in particular, although the system generates a ranked list of possible interpretations, we report numbers based on the top answer only (Precision@1). The results are presented in Table 4. In short, Bio-SODA performs second for the QALD4 dataset, while it significantly outperforms the state-of-the-art systems for the other two datasets by an F1-score increase of 46% and 33%, respectively. For easy accessibility to Bio-SODA, as well as reproducibility of the results, we also provide a demo page for each of the three datasets, available online[16].

[12] e.g. http://wdaqua-frontend.univ-st-etienne.fr/

[13] A live demo can be tested with any SPARQL endpoint at http://www.irisa.fr/LIS/ferre/sparklis/

[14] See http://cs-gw.utcluj.ro/~anca/GFMed/index.html

[15] Available at https://github.com/AskNowQA/SQG/

[16] Bio-SODA Demo: http://biosoda.expasy.org/welcome/

We will now discuss the performance of each of measured systems in more detail.

System/Stats	Precision	Recall	F1
QALD4			
GFMed	1	0.99	0.99
SQG	0.42	0.42	0.42
Sparklis (5.5 steps/query)	0.88	0.88	0.88
Bio-SODA	0.61	0.60	0.60
Bioinformatics			
GFMed	0	0	0
SQG	0.16	0.16	0.16
Sparklis	-	-	-
Bio-SODA	0.6	0.6	0.6
CORDIS			
GFMed	0	0	0
SQG	0.33	0.33	0.33
Sparklis (6.2 steps/query)	1	1	1
Bio-SODA	0.66	0.66	0.66

Table 4

Evaluation results. By considering a perfect user of the Sparklis tool, the minimum number of manual steps for composing a query (averaged over all queries) is shown between parentheses.

GFMed, the highest scoring in the QALD4 challenge, cannot (nor was it intended to) be used outside this dataset without rewriting the set of grammar rules that are strictly designed for question answering over specific releases of *Diseasome*, *Drugbank* and *Sider*.

SQG on the other hand, originally evaluated on the LC-Quad [50] benchmark, does not support complex multi-hop questions, nor filters or queries involving literals. "*Show me projects which started in **2020**?*" is an example of such a query, where *2020* is a numerical literal, as opposed to a linkable entity. While in the case of LC-Quad these limitations do not impact performance, all three datasets considered in our evaluation include such features, which explains the poorer performance of SQG: an F1-score of 0.42 in the case of QALD4, only 0.33 in the CORDIS dataset, and finally 0.16 in the case of the bioinformatics dataset. We note that these results are a theoretical best, since for SQG we assume perfect entity and property linking, leading to the highest performance it can achieve.

Finally, Sparklis is not a question answering system per-se, but rather a query builder, which helps users form the correct question by composing building blocks starting from examples of class names, properties, values etc. Therefore, in order to answer questions, we needed to rephrase them from the available building blocks *manually*. On the positive side, we found Sparklis to be a powerful system, because it enables building a rich variety of query types out-of-the-box. To achieve this, only the SPARQL endpoint URL of the target RDF data store is required. Using the query building methodology of Sparklis, 44 out of 50 questions in the QALD4 biomedical benchmark can be answered. Furthermore, all questions in the CORDIS dataset can also be answered. Although this result might seem surprising, recall that the major challenge of this dataset is disambiguation. The manual query building process in Sparklis addresses exactly this problem, provided that the user knows very well how the data are structured and semantically represented. Therefore, on the negative side, we found that the query building methodology requires precise understanding of the data model, especially if multiple classes have the same label, as is the case in QALD4. For example, answering the question *Which drugs might lead to strokes?* requires knowing that the *Drugs* class to be used is the one in *Sider*, as opposed to the one in *Diseasome*. Furthermore, formulating questions in Sparklis is a manual and therefore time-consuming process. Even when making the strong assumption that the user has perfect knowledge of the data model, as well as of the features of Sparklis (for example, how to correctly formulate aggregations, which can be challenging), the minimal number of manual steps required to formulate questions is on average 5.5 interactions per question for QALD4 and 6.2 for CORDIS, with a maximum of 10 for the more complex questions. In most cases, the question resulting from composing the building blocks will be significantly different from a true natural language question. We did not pursue this approach on the bioinformatics dataset, because complex concepts in this dataset (ortholog, paralog) cannot be expressed through the query building mechanism. More precisely, Sparklis does not support complex property paths such as the ones previously introduced in Table 2.

Bio-SODA is a middle-ground between the generic, but manual approach of Sparklis, and the grammar-based approach of GFMed, which is not easily transferable to a new domain. More precisely, Bio-SODA achieves relatively good performance (around 0.6 F1 score) across the three datasets without requiring manual intervention. The only exception are two custom rules for the bioinformatics dataset, as described in Table 2. These help answer 4 out of 30 queries. Although GFMed has the best results for QALD4, it cannot be used outside this dataset without a complete rewriting of the grammar rules. Sparklis also achieves better re-

sults on the two datasets tested, but with the big disadvantage that it is a manual approach, where the user must understand the data model in order to compose questions correctly.

Our findings are further detailed in the Evaluation folder in our GitHub repository. We provide an analysis of the results for Bio-SODA, and highlight limitations of the current prototype, in the following sections.

5.4. Impact of Ranking Algorithm

We conducted an ablation study to quantify the importance of ranking by PageRank score of candidate matches. For this purpose, we disable our ranking algorithm and instead use a simple string similarity-based ranking algorithm for candidate matches, returning the *overall* minimal subgraph as the top answer.

The results, displayed in Table 5, show that for the CORDIS dataset in particular, ranking makes a crucial difference. The reason for this is that for most of the keywords that describe metadata (such as class names, like *Project Topic* or *Subject Area*), there exists in the dataset a project whose acronym matches exactly. For example, there exist projects with acronyms such as *Topic*, *Area*, *Host*, *Code*, which are (according to string similarity only) classified as best matches for tokens in the original question. Constructing the *overall* minimal subgraph leads to wrong results in almost all cases, except for only 3 out of 30 questions, where there is no ambiguity. Note that adding *no other change* aside from considering PageRank scores in ranking enables answering 17 more queries out of 30 for this dataset.

Dataset	(a) Correct with Bio-SODA Ranking	(b) Correct with String Similarity Ranking
QALD4	30/50	23/50
Bioinformatics	18/30	12/30
CORDIS	**20/30**	**3/30**

Table 5

Ablation study: (a) ranking with node centrality measure versus (b) traditional ranking approach with string similarity and overall minimal subgraph as top result.

5.5. Error Analysis and Remaining Problems

In the QALD4 biomedical benchmark, Bio-SODA correctly answered 30 out of 50 questions with an additional 2 partially correct. We note that 1 question in QALD4 cannot be answered by Sparklis nor Bio-SODA due to missing label information. More precisely, the instance <http://www4.wiwiss.fu-berlin.de/diseasome/resource/genes/EDNRB> is the target of the question “Which genes are associated with Endothelin receptor type B?". However, the label *Endothelin receptor type B* is not assigned in the official dataset of the benchmark, nor can it be derived from the URI fragment, for example. Upon closer inspection, the question is actually ill-formulated. Since *EDNRB* itself is a gene, the correct question should be “Which *diseases* are associated with EDNRB?". In total, we have found at least 4 out of 50 entries in the dataset to contain errors, either in the question formulation, or in the ground truth answer. These have already been discussed in previous studies [49].

An additional number of questions cannot be answered by Bio-SODA across the three datasets due to other reasons. We summarise them in Figure 5, explained in the following:

- *Aggregations*. Our system currently does not support questions that require aggregations, such as *Count*, *Sum* etc. An example such question would be *Count the projects in the life sciences domain*. A possible solution to this would be to include pre-defined patterns or training a question classifier for this purpose.
- *Superlatives/Comparatives*. Another unsupported feature in the current prototype is the use of quantifiers (superlatives or comparatives). An example would be *Which drug has the highest number of side-effects?* The main reason for the lack of support in this feature is that determining the property which should be quantified requires either using the dependency parse tree (in which case the questions should be grammatically correct) or using heuristics, which may be error-prone. The requirement to support keyword searches in Bio-SODA made it challenging to take into account the dependency parse tree for questions, given that the performance of dependency parsers decreases with missing words (such as keyword searches) [26]. One possible solution for this would be to only use Part of Speech (PoS) tagging to detect superlatives, considering the dependency parse tree when these are present. The direction of the superlative (*e.g. most* as opposed to *least*) would need to be detected via a classifier.
- *Conjunctions*. Conjunctive questions which involve multiple instances of the same class are not supported in the current prototype. An example

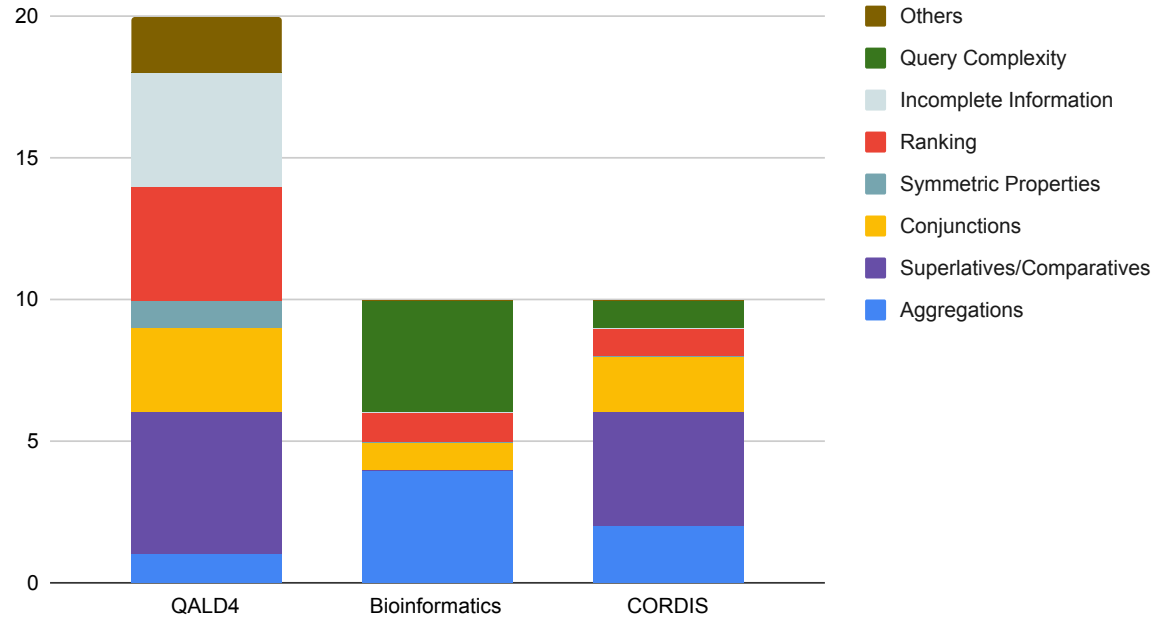

Fig. 5. Bio-SODA failure analysis. Out of the total 50 questions in the QALD4 biomedical benchmark, Bio-SODA cannot correctly answer 20. A further 12 out of 30 cannot be answered in the bioinformatics dataset, mainly due to query complexity (some queries having more than 10 triple patterns). Finally, on the CORDIS dataset 10 out of 30 queries cannot be answered, a large fraction of which include features currently unsupported in Bio-SODA: aggregations, superlatives and comparatives, conjunctions etc.

of such a case is *List drugs that lead to strokes and arthrosis*. This limitation derives from our methodology in computing the minimal subgraph covering candidate matches, which would require special handling for cases when multiple candidates of the same class are present in a question.

– *Properties with same domain and range*. Stemming from the same limitation mentioned above, these properties are currently not supported. In QALD4, the only instance of this is the *diseaseSubtypeOf* property, which has as both domain and range the *Disease* class. In the bioinformatics dataset we handle symmetric properties describing *ortholog* and *paralog* genes through the custom rules exemplified in Table 2.
– *Ranking*. One of the major sources of failure in our prototype remains ranking. In the QALD4 dataset, ranking problems affect 4 out of 50 queries. An example is: *What are the diseases caused by Valdecoxib?*. Here, the system cannot correctly choose *Drug - sideEffect - Side_Effect* over the alternative *Disease - possibleDrug - Drug*. The reason for this is that the *Disease* class matches exactly the term in the question, while the *Drug* class in *Diseasome* has a higher PageRank score than the one in *Sider*. Furthermore, the combination of these two candidate matches also leads to the smallest subgraph. A possible solution to the problem would be incorporating word embeddings or synonyms in the indexing phase, or rephrasing the question. Despite this issue, the user is still able to identify the correct interpretation among the lower ranked answers, based on the explanations provided in the output.

A more complex corner-case is part of the bioinformatics dataset, *What are the genes with lung in the description?* The term *lung* is commonly used to refer to an *Anatomical Entity*. This also reflects in the node importance of this match in the dataset. Therefore, the system cannot correctly determine that, in the context of this question, it should instead be considered part of the *description* property of a *Gene*. The correct candidate match scores very low, resulting in the correct answer also being ranked too low. A similar example from QALD4 is *Which drugs have bipolar disorder as indication?*, where *bipolar disorder* is matched against a *Disease* instead of a drug indi-

cation. In these cases user disambiguation, at the level of candidate matches, would solve the problem.

– *Incomplete information*. This problem affects mainly the results in the QALD4 dataset, more precisely 4 out of 50 queries. We have already covered the example of the question targeting the *EDNRB* gene, which lacks the correct label in the official dataset. We currently do not enrich the inverted index with synonyms or external information, which means questions must be formulated in terms of the available vocabulary of the dataset. An additional three questions cannot be answered because they refer to URIs that do not have any class defined in the data, therefore the system cannot attach the candidate matches anywhere in the Summary Graph. An example is the *drugType* property, which can take two values, either *http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugtype/experimental* or *http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugtype/approved*. We believe a better modelling of the data should have provided, for example, either these as a *xsd:anyURI* datatype, given they are not used for any other purposes, or defined some class for both.
– *Query complexity*. The bioinformatics dataset covers queries with high complexity, which are difficult to solve especially since they include symmetric properties, with multiple instances of the same class, each filtered according to different conditions. An example of such a question is *Retrieve Oryctolagus cuniculus' proteins encoded by genes that are orthologous to Mus musculus' HBB-Y gene*. Here, the task is to retrieve *Gene* instances in a particular *Taxon* (species), namely the rabbit (*Oryctolagus cuniculus*), which are *orthologs* (symmetric property) of a second instance of *Gene*, labeled *HBB-Y*, in a different species, namely the mouse (*Mus musculus*). The resulting query has over 15 triple patterns, with 3 filters (the 2 species names plus the gene name). Bio-SODA currently cannot answer questions which involve multiple instances of the same class with additional constraints on each instance. This is particularly challenging to solve without considering the dependency parse tree. Therefore, we currently only support generic questions, such as *What are the orthologs of Mus musculus' HBB-Y gene?*
– *Others*. Two questions in the QALD4 dataset have particular challenges, the first being a stemming error. In the question *Give me drugs in the gaseous state*, the term *gaseous* cannot be correctly stemmed to *gas*. The second type of error is due to unsupported *ASK* queries, *e.g. Are there drugs that target the Protein kinase C beta type?*. Here, Bio-SODA retrieves examples of such drugs, instead of the boolean *True*. However we do not consider this a fundamental limitation and a question type classifier could be added in future work.

We report a more detailed analysis of all systems considered in this paper in the Evaluation folder in our GitHub repository.

6. Lessons Learned

6.1. Design Goals for Question Answering Systems over Domain Knowledge Graphs

Stemming from the challenges of closed-domain question answering, which we introduced in Section 2, as well as our experience in using Bio-SODA across several real-world datasets, we derived the following design goals for question answering systems over domain knowledge graphs:

– *Generality*: The system should be easily adaptable to new datasets. In particular, the system should be able to answer questions in a new domain without manual intervention and without relying on extensive training data, which is hard to obtain in many domains. In this line, a desirable property is also the ability to cope with "real-world" datasets, dealing with incompleteness in the data, in the form of:
  * missing schema information (should be inferred from instance-level data);
  * missing labels (should be incorporated from URIs whenever meaningful);
  * other issues; an example is the owl:sameAs property, which is an equivalence relationship, however, in many cases also has a defined directionality. More precisely, an RDF store can assert "A owl:sameAs B" without also asserting "B owl:sameAs A". When "owl:sameAs" inferences are not derived by Web Ontology Language (OWL) reasoners [11], this can have

unexpected consequences, given that writing the wrong SPARQL query (*i.e.* wrong directionality) will lead to empty results. This is a known issue in the the QALD4 biomedical dataset [24]. One possible solution is to always consider the UNION of both directions: "A owl:sameAs B" UNION "B owl:sameAs A" in constructing SPARQL queries.

- *Extensibility*: The system should easily work with multiple datasets (provided they are already semantically aligned—*i.e.*, data integration is a prior requirement). Many studies introduce possible approaches for data integration, including a recent approach for ontology-based data integration, covering one of the bioinformatics use cases presented in this paper [47].
- *Configurability*: The database owner should be able to specify properties (*e.g.* labels, descriptions) should be searchable using the system. Our experience with real-world datasets showed that in general it is not desirable for *all* properties to be indexed. As an example, in many cases, fields in the queried data sources can be either redundant or too verbose. In bioinformatics, these are abstracts of papers that are assigned as values to an RDF property, whose length can therefore be up to 300 words. Similarly, in the CORDIS dataset, these are the abstracts of the EU projects. These cases should be handled through a dedicated approach, for example, based on classical information retrieval methods (see an example in [37]).

- *Explainability*: The system should clearly guide the user through how a question was processed and interpreted. This starts from explaining which concepts were matched in relation to the original question, continuing with how these candidate matches are composed together in a query graph in order to provide the final SPARQL query. Finally, the query results should be understandable as well. Therefore, the projected variable names should be meaningful. Furthermore, labels and descriptions should be part of the result set, as opposed to only URIs, which in most cases do not contain human-readable information. A concrete example of two possible interpretations for the question "Which drugs lead to strokes?" are shown in Table 6. The first interpretation given on the top is correct, but hard to validate since no labels are shown. The second interpretation is easier to understand given that the SPARQL query includes meaningful variable names and human-readable information (e.g. labels) in the result set.

7. Related Work

The problem of question answering over structured data has been well-studied in recent years, with a growing number of published systems, particularly in open-domain question answering. Recent surveys on natural language interfaces to databases include [2, 9].

In parallel, the biomedical field has seen a growth of dedicated systems for question answering. Examples include GFMed [34] and Pomelo [25] – the two highest ranked systems in the QALD4 biomedical challenge – as well as more recent systems [24]. However, these can generally be considered as expert systems, with lower generalizability to other domains, given that they extensively rely on manually handcrafted rules, which often depend on domain expertise.

Our work aims to bridge the gap between the two parallel efforts by solving the *common case* in a domain-independent manner. For this, Bio-SODA relies on a generic graph-based approach in order to generate a ranked list of candidate SPARQL queries from a given question. We enable the addition of custom rules only for *corner cases* when needed.

Many recent KGQA systems [13, 53] have been evaluated using the LC-Quad benchmark of 5000 questions over DBpedia [50]. Although this benchmark is an important step forward, particularly for enabling machine learning approaches, it does not cover complex multi-hop questions, which makes it unclear how the results would generalize to this case. As an example, at the time of writing, the current implementation of SQG, which is publicly available [55], would not work for complex question answering on a new knowledge graph without significant changes to the code base, as it is targeted at question answering over DBpedia and more specifically in the format required by the LC-Quad benchmark. We recall here that LC-Quad includes queries with a complexity of at most two hops, which is also reflected in the implementation of SQG at the time of this writing. ExSQG, an extension of SQG, was recently proposed [1] to support richer query types. Although ExSQG supports ordinal questions (*e.g.*, superlatives), it does not support string-based filters, which Bio-SODA makes extensive use of.

1) A simplified SPARQL query could be:
Query: SELECT ?x where { ?x sider:sideEffect side_effects:C0038454.}
Results: ?x=sider_drug:163742; ?x=sider_drug:4440;

2) A more meaningful and interpretable result set would be given by:

Query:
```
SELECT * where {
?drug sider:sideEffect ?side_effect.
?drug rdfs:label ?drug_label.
?side_effect sider:sideEffectName ?side_effect_name.
FILTER(contains(?side_effect_name), "stroke"). }
```

	?drug	**?drug_label**	?side_effect	**?side_effect_name**
Results:	sider_drug:163742	"Arixtra"	side_effects:C0038454	"stroke"
	sider_drug:4440	"Amerge"	side_effects:C0038454	"stroke"

Table 6

Example showing two equivalent interpretations for the question "Which drugs lead to strokes?". The namespace URIs are omitted for readability and replaced with prefixes (e.g. "sider:") The first interpretation returns correct results, however, they are hard to validate by a non-expert user, given that the response only contains URIs (essentially, opaque numerical identifiers). In contrast, in the second variant, the system provides a more understandable SPARQL query, with meaningful variable names, but more importantly a more comprehensive result set. Note that the human readable name of an instance returned in the result is not necessarily always found in the label property (e.g. sider:sideEffectName).

More recent KGQA systems, such as [14, 53], support multiple knowledge graphs, but are limited to queries with a complexity of at most three triple patterns. Similarly, existing end-to-end QA systems, based on machine learning approaches, such as [32], can only handle simple questions. These approaches have the added drawback that they only generate a single answer, as opposed to multiple candidates. Furthermore, end-to-end approaches suffer from the problem of lack of explainability, which makes it challenging for users to validate the correctness of the result. Explainability has therefore in this context become an active area of research, with solutions proposed including translating back structured queries into natural language sentences [12, 29, 38] or summarizing the entities in the results [16].

Disambiguation is one of the major tasks of question answering systems. One possible solution for this is to limit the interface to a controlled natural language and involve the user in constructing questions from the available building blocks. Sparklis [21] is a query building system that enables answering controlled natural language questions over knowledge graphs out-of-the-box. However, this process is manual and therefore time-consuming, which makes it less convenient than a true natural language interface.

One of the systems closest to ours is the KG-agnostic WDAqua-core1 [13]. The system supports multiple knowledge bases in several languages. However, the system is only available as a demo. Although the authors mention that node relevance can in principle be taken into account in ranking, it is not clear whether the approach was used in the evaluation or whether the ranking function was learned based on training data. Furthermore, the system identifies relations from a user question based on a dictionary and does not use word embeddings for filtering out spurious candidate matches. One of the strong points of the system is its portability, which is demonstrated across 5 datasets. We have also tested Bio-SODA across 3 multi-domain datasets, achieving satisfactory results across all three benchmarks.

8. Conclusions and Outlook

In this paper we have introduced Bio-SODA, a question answering system for domain knowledge graphs, which we evaluated across three real-world datasets pertaining to different domains: biomedical, orthology and gene expression, and finally EU-funded projects. Our results have shown that Bio-SODA outperforms state-of-the-art systems that are publicly available for testing by a 20% F1-score improvement and more. The main advantage of Bio-SODA over existing open-source systems is that it can handle *complex, multi-triple pattern queries* without requiring user guidance and training data. Bio-SODA uses a novel ranking approach that takes into account both string and semantic similarity, as well as node centrality of candidate

matches. Our experiments demonstrate that our ranking approach improves the quality of results, particularly in the context of datasets which can suffer from redundancy and imprecise labels.

However, making question answering systems for domain knowledge graphs work in the real-world will require a multi-way dialog between disciplines, involving the database community, the natural language processing community and domain experts. Although there is clearly a long remaining road ahead, our work aims at focusing the discussion on domain knowledge graphs, such as scientific datasets in RDF, as well as their specific challenges.

As a first step in future work, we plan to add user feedback to the question answering process by involving the user in a disambiguation dialog for selecting the best candidate matches. We also plan to consider the users' feedback for ranking the best answer among resulting candidate queries. As a long term direction for future research, we envision compiling a benchmark of cross-domain question-answer pairs, similarly to the Spider benchmark in the relational database world [54], which would enable research into refining pre-trained KGQA models for new domains.

9. Acknowledgements

We thank the Swiss National Science foundation for funding (NRP 75, grant 407540_167149), Lukas Blunschi for the implementation of the SODA system for keyword search system over relational databases [7], on which our prototype is based, and Katrin Affolter for important contributions to the natural language processing pipeline in Bio-SODA.

References

[1] A. Abdelkawi, H. Zafar, M. Maleshkova, and J. Lehmann. Complex query augmentation for question answering over knowledge graphs. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 571–587. Springer, 2019.

[2] K. Affolter, K. Stockinger, and A. Bernstein. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819, 2019.

[3] A. M. Altenhoff, N. M. Glover, C.-M. Train, K. Kaleb, A. Warwick Vesztrocy, D. Dylus, T. M. de Farias, K. Zile, C. Stevenson, J. Long, et al. The oma orthology database in 2018: retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces. *Nucleic acids research*, 46(D1):D477–D485, 2018.

[4] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[5] F. B. Bastian, J. Roux, A. Niknejad, A. Comte, S. Fonseca Costa, T. M. de Farias, S. Moretti, G. Parmentier, V. R. de Laval, M. Rosikiewicz, J. Wollbrett, A. Echchiki, A. Escoriza, W. H. Gharib, M. Gonzales-Porta, Y. Jarosz, B. Laurenczy, P. Moret, E. Person, P. Roelli, K. Sanjeev, M. Seppey, and M. Robinson-Rechavi. The Bgee suite: integrated curated expression atlas and comparative transcriptomics in animals. *Nucleic Acids Research*, 10 2020. gkaa793.

[6] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette. Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, 41(5):706–716, 2008.

[7] L. Blunschi, C. Jossen, D. Kossmann, M. Mori, and K. Stockinger. Soda: Generating sql for business users. *Proceedings of the VLDB Endowment*, 5(10):932–943, 2012.

[8] A. Bonifati, W. Martens, and T. Timm. An analytical study of large sparql query logs. *The VLDB Journal*, pages 1–25, 2019.

[9] N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer. Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*, 2019.

[10] T. M. de Farias, H. Chiba, and J. T. Fernández-Breis. Leveraging logical rules for efficacious representation of large orthology datasets, 2017.

[11] T. M. [de Farias], A. Roxin, and C. Nicolle. Swrl rule-selection methodology for ontology interoperability. *Data & Knowledge Engineering*, 105:53 – 72, 2016. Knowledge Engineering for Enterprise, Integration, Interoperability and Networking: Theory and Applications.

[12] D. Deutch, N. Frost, and A. Gilad. Explaining natural language query results. *The VLDB Journal*, 29(1):485–508, 2020.

[13] D. Diefenbach, A. Both, K. Singh, and P. Maret. Towards a question answering system over the semantic web. *Semantic Web*, (Preprint):1–19, 2018.

[14] D. Diefenbach, J. Giménez-Garcıa, A. Both, K. Singh, and P. Maret. Qanswer kg: Designing a portable question answering system over rdf data. 2020.

[15] D. Diefenbach, K. Singh, and P. Maret. Wdaqua-core1: a question answering service for rdf knowledge bases. In *Companion Proceedings of the The Web Conference 2018*, pages 1087–1091, 2018.

[16] D. Diefenbach and A. Thalhammer. Pagerank and generic entity summarization for rdf knowledge bases. In *European Semantic Web Conference*, pages 145–160. Springer, 2018.

[17] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International Semantic Web Conference*, pages 69–78. Springer, 2019.

[18] M. Dumontier, A. Callahan, J. Cruz-Toledo, P. Ansell, V. Emonet, F. Belleau, and A. Droit. Bio2rdf release 3: a larger connected network of linked data for the life sciences. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track*, volume 1272, pages 401–404, 2014.

[19] B. Ell, D. Vrandečić, and E. Simperl. Labels in the web of data. In *International Semantic Web Conference*, pages 162–176. Springer, 2011.

[20] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628, 2010.

[21] S. Ferré. Sparklis: an expressive query builder for sparql endpoints with guidance in natural language. *Semantic Web*, 8(3):405–418, 2017.

[22] K. Gkirtzou, K. Karozos, V. Vassalos, and T. Dalamagas. Keywords-to-sparql translation for rdf data search and exploration. In *International Conference on Theory and Practice of Digital Libraries*, pages 111–123. Springer, 2015.

[23] S. Hakimov, C. Unger, S. Walter, and P. Cimiano. Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In *International Conference on Applications of Natural Language to Information Systems*, pages 103–109. Springer, 2015.

[24] T. Hamon, N. Grabar, and F. Mougin. Querying biomedical linked data with natural language questions. *Semantic Web*, 8(4):581–599, 2017.

[25] T. Hamon, N. Grabar, F. Mougin, and F. Thiessard. Description of the pomelo system for the task 2 of qald-2014. *CLEF (Working Notes)*, 1212:28, 2014.

[26] H. B. Hashemi and R. Hwa. An evaluation of parser robustness for ungrammatical sentences. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1765–1774, 2016.

[27] A. Hasnain, Q. Mehmood, S. S. e Zainab, M. Saleem, C. Warren, D. Zehra, S. Decker, and D. Rebholz-Schuhmann. Biofed: federated query processing over life sciences linked open data. *Journal of biomedical semantics*, 8(1):13, 2017.

[28] K. Kellou-Menouer and Z. Kedad. Schema discovery in rdf data sources. In *International Conference on Conceptual Modeling*, pages 481–495. Springer, 2015.

[29] A. Kokkalis, P. Vagenas, A. Zervakis, A. Simitsis, G. Koutrika, and Y. Ioannidis. Logos: a system for translating queries into narratives. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 673–676, 2012.

[30] F. Li and H. Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.

[31] F. Li and H. Jagadish. Understanding natural language queries over relational databases. *ACM SIGMOD Record*, 45(1):6–13, 2016.

[32] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220, 2017.

[33] G. Maheshwari, P. Trivedi, D. Lukovnikov, N. Chakraborty, A. Fischer, and J. Lehmann. Learning to rank query graphs for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 487–504. Springer, 2019.

[34] A. Marginean. Question answering over biomedical linked data with grammatical framework. *Semantic Web*, 8(4):565–580, 2017.

[35] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011.

[36] S. Moen and T. S. S. Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of LBM*, pages 39–44, 2013.

[37] S. Nadig, M. Braschler, and K. Stockinger. Database search vs. information retrieval: A novel method for studying natural language querying of semi-structured data. In *International Conference on Language Resources and Evaluation (LREC)*, 2020.

[38] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber. Sorry, i don't speak sparql: translating sparql queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*, pages 977–988, 2013.

[39] A. Olieman, H. Azarbonyad, M. Dehghani, J. Kamps, and M. Marx. Entity linking by focusing dbpedia candidate entities. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 13–24, 2014.

[40] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[41] H. Paulheim and C. Bizer. Type inference on noisy rdf data. In *International semantic web conference*, pages 510–525. Springer, 2013.

[42] N. Redaschi, U. Consortium, et al. Uniprot in rdf: Tackling data integration and distributed annotation with the semantic web. *Nature precedings*, pages 1–1, 2009.

[43] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan. Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, 9(12):1209–1220, 2016.

[44] A. Sakor, K. Singh, and M.-E. Vidal. Falcon: An entity and relation linking framework over dbpedia. 2019.

[45] M. Saleem, S. N. Dastjerdi, R. Usbeck, and A.-C. N. Ngomo. Question answering over linked data: What is difficult to answer? what affects the f scores? In *BLINK/NLIWoD3@ ISWC*, 2017.

[46] S. Shekarpour, E. Marx, A.-C. N. Ngomo, and S. Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 30:39–51, 2015.

[47] A. C. Sima, T. Mendes de Farias, E. Zbinden, M. Anisimova, M. Gil, H. Stockinger, K. Stockinger, M. Robinson-Rechavi, and C. Dessimoz. Enabling semantic queries across federated bioinformatics databases. *Database*, 2019, 2019.

[48] K. Singh, I. Lytra, A. S. Radhakrishna, S. Shekarpour, M.-E. Vidal, and J. Lehmann. No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph. *arXiv preprint arXiv:1809.10044*, 2018.

[49] D. Song, F. Schilder, C. Smiley, C. Brew, T. Zielund, H. Bretz, R. Martin, C. Dale, J. Duprey, T. Miller, et al. Tr discover: A natural language interface for querying and analyzing interlinked datasets. In *International Semantic Web Conference*, pages 21–37. Springer, 2015.

[50] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 210–218. Springer, 2017.

[51] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):138, 2015.

[52] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (qald-4). 2014.

[53] S. Vakulenko, J. D. Fernandez Garcia, A. Polleres, M. de Rijke, and M. Cochez. Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1431–1440, 2019.

[54] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

[55] H. Zafar, G. Napolitano, and J. Lehmann. Formal query generation for question answering over knowledge bases. In *European Semantic Web Conference*, pages 714–728. Springer, 2018.

[56] W. Zheng, J. X. Yu, L. Zou, and H. Cheng. Question answering over knowledge graphs: question understanding via template decomposition. *Proceedings of the VLDB Endowment*, 11(11):1373–1386, 2018.

[57] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324, 2014.