Semantic Web 0 (0) 1
IOS Press

Querying Biomedical Linked Data with Natural Language Questions

Thierry Hamon [a,b,*], Natalia Grabar [c], Fleur Mougin [d]
[a] *LIMSI-CNRS, Orsay, France*
E-mail: hamon@limsi.fr
[b] *Université Paris 13, Sorbonne Paris Cité, France*
[c] *STL UMR8163 CNRS, Université Lille 3, France*
E-mail: natalia.grabar@univ-lille3.fr
[d] *Université Bordeaux, ERIAS, Centre INSERM U897, ISPED, France*
E-mail: fleur.mougin@u-bordeaux.fr

Abstract. A recent and intensive research in the biomedical area enabled to accumulate and disseminate biomedical knowledge through various knowledge bases increasingly available on the Web. The exploitation of this knowledge requires to create links between these bases and to use them jointly. Linked Data, SPARQL language and interfaces in Natural Language question-answering provide interesting solutions for querying such knowledge bases. However, while using biomedical Linked Data is crucial, life-science researchers may have difficulties using SPARQL language. Interfaces based on Natural Language question-answering are recognized to be suitable for querying knowledge bases. In this paper, we propose a method for translating natural language questions into SPARQL queries. We use Natural Language Processing tools, semantic resources and RDF triples description. We designed a four-step method which allows to linguistically and semantically annotate questions, to perform an abstraction of these questions, then to build a representation of the SPARQL queries, and finally to generate the queries. The method is designed on 50 questions over three biomedical knowledge bases used in the task 2 of the QALD-4 challenge framework and evaluated on 27 new questions. It achieves good performance with 0.78 F-measure on the test set. The method for translating questions into SPARQL queries is implemented as a Perl module and is available at `http://search.cpan.org/~thhamon/RDF-NLP-SPARQLQuery/`.

Keywords: Natural Language Processing, SPARQL, biomedical domain, semantic resources

1. Introduction

A recent and intensive research in the biomedical area enabled to accumulate and disseminate biomedical knowledge through various knowledge bases (KBs) increasingly available on the Web. Such life-science bases usually focus on a specific type of biomedical information: clinical studies in ClinicalTrials.gov[1]; drugs and their side effects in Sider [16]; chemical, pharmacological and target information on drugs in DrugBank [32], etc.

Nowadays, creating connections between these KBs is crucial for obtaining a more global and comprehensive view on the links between different biomedical components. Such links are also required for inducing and producing new knowledge from the already available data. There is a great endeavour in the definition of Open Linked Data to connect such knowledge and in taking advantage of the SPARQL language to query multiple KBs jointly. Particularly, the creation of fine-grained links between the existing KBs related to

*Corresponding author. E-mail: hamon@limsi.fr.
[1] `http://clinicaltrials.gov/`

drugs is a great challenge that is being addressed, for instance, by the project Linked Open Drug Data (LODD)[2]. The knowledge recorded in the knowledge bases and dataset interlinks are represented as RDF triples, on the basis of which linked data can then be queried through a SPARQL endpoint. However, typical users of this knowledge, such as physicians, life-science researchers or even patients, cannot manage the syntactic and semantic requirements of the SPARQL language neither can they manage the structure of various knowledge bases. This situation impedes an efficient use of knowledge bases and retrieval of useful information. Therefore, it is important to design friendly interfaces that mediate technical and semantic complexity of the task and provide simple approaches for querying knowledge bases. The main challenge is then to design optimal methodologies for an easy and reproducible rewriting of natural language questions into SPARQL queries. In the remaining of this work, the term *question* means the natural language expressions uttered by human users to formulate their information need, while the term *query* designates the same expression formalised with the SPARQL syntax and semantics.

We start with the presentation of some related works.

2. Related Work

Querying Linked Data requires to define the end user interfaces which hide the underlying structure of the KB as well as the SPARQL syntax. Usually, three ways are identified for querying Linked Data: Knowledge-Based Specific Interface, Graphical Query Builder and Question-Answering System [12]. In this work, it has also been demonstrated that Natural Language interfaces are the most suitable: indeed, for querying KBs and Semantic Web data, the use of full and standard sentences is preferred to the use of keywords, menus or graphs [12].

Another important distinction is related to the types of linked data which are processed (typically, general [5][15][31] or specialized [1] languages KBs) and the purpose of this processing. Concerning the purpose, two kinds of work can be distinguished: (1) transformation of natural language questions into SPARQL queries; (2) transformation of natural language questions into SPARQL queries and questioning KBs. We first present works that address only the transformation of natural language questions into SPARQL queries (Sect. 2.1). We then present those works that go beyond and, in addition, querry KBs and evaluate the obtained answers (Sect. 2.2). Finally, we also present other related research questions addressed in the work (Sect. 2.3).

2.1. Transformation of Questions into Queries

The main objective of this kind of works is to propose methods for a more efficient transformation of natural language questions into SPARQL queries. Most of the existing approaches rely on patterns or templates.

One Question-Answering system (AutoSPARQL) is based on active supervised machine learning independent from the KB [18]. The SPARQL query model is learnt from Natural Language questions. The authors report that 50 questions are successfully transformed with the system. Another method is based on modular patterns for parsing the questions [25]. Semantic relations are identified on the basis of the first keywords detected. The method is tested on 160 movie-related questions. One advantage is that the method requires only four modular query patterns, while in a previous work of the authors twelve patterns were necessary [24].

The use of resources automatically derived from ontologies or KBs also provides the possibility to transform questions into formal queries, such as those defined using the SeRQL language [30]: questions undergo a set of treatments (e.g. linguistic analysis, string similarity computing). Applied to a set of 22 questions, the method can interpret and transform correctly 15 questions (68%).

Existing tools can also be utilized. For instance, a multilingual toolkit available for 36 languages [26] has been used for the transformation of questions into queries [6]. Correspondences between linguistic units and SPARQL elements are established: common noun (*Kind*), noun phrase (*Entity*), verb phrase (*Property*) and verb phrase with a higher arity (*Relation*). The evaluation is performed on seven languages. The results indicate

[2] `http://www.w3.org/wiki/HCLSIG/LODD`

that up to 112 basic query patterns are to be used and can be combined with several logical operators.

Finally, a manually-written grammar together with ontological knowledge allow processing 145 questions out of 164 (88% coverage) [9].

2.2. Query Generation and Questioning over Linked Data

The main objective of the related set of works is more complex than in works presented in Sect. 2.1: first, the system has to transform the natural language questions into SPARQL queries; and second, it has to question the KB in order to get the best results possible when querying the linked data. The main advantage of this kind of works is that they cover the entire querying process. Moreover, they allow to evaluate the final results (answers extracted from the KBs) and to provide precise evaluation figures. Often, NLP tools and methods are used for the transformation of questions into queries. We propose three such experiments.

In one study, the system is template-based and relies on NLP tools and semantic resources [31]. The application of the system on 50 questions from DBpedia proposed by the QALD-2 challenge gives competitive results with an average of 0.62 F-measure obtained with 39 questions (the average recall is 0.63 and the average precision is 0.61), but shows a low coverage because 11 questions are not covered by the templates.

Another study relies on a hybrid approach [1]: a SVM machine learning-based approach, which is used to extract the characteristics of the questions (named entities, relations), is combined with patterns to generate the SPARQL queries. The method is applied to medical questions issued from a journal. The evaluation is carried out on 100 questions and the corresponding queries are tested on clinical documents. The method achieves a 0.62 precision when querying the documents.

Finally, an approach based on a knowledge-driven disambiguation of questions and on a coloured activation of query graph [20] has been tested on 100 questions from the QALD-3 dataset. According to tested settings, the system F-measure varies from 0.4 (QALD-3 dataset) to 0.6 with the entity search, and up to 0.8 with a boolean setting. Among the difficulties observed, the authors notice errors due to the relation interpretation, missing lexical knowledge, parsing of complex questions and remaining difficulties with ambiguities.

Notice that recently, the Question Answering over Linked Data (QALD-4) challenge proposes a task[3] dedicated to the retrieval of precise biomedical information in linked KBs with questions expressed in natural language.

2.3. Other Related Research Questions

Several research questions can be related to the querying of linked data through the natural language interfaces. Usually this kind of work aims at improving specific points: identification of different types of SPARQL queries (select, construct, ask, describe) [19], detection of named entities [14], generation of SPARQL templates [13], classification of semantic correspondences between question units and query elements [10], design of a SPARQL solver based on contraint programming to query RDF documents [17], processing of complex queries and their decomposition into subqueries [23].

3. Objectives

The objective of our work is to propose a novel and end-to-end method for translating natural language questions into SPARQL queries and for querying KBs. The method is based on the use of Natural Language Processing (NLP) tools and resources for enriching question with linguistic and semantic information. Questions are then translated into SPARQL with a rule-based approach.

Our method is close to those presented in Sect. 2.2. In comparison with the closest work [31] identified from the state of the art, the main difference is that we use information issued from the Linked Data resources to semantically annotate the questions and to define the frames (i.e., linguistic representations of the RDF schema) in order to model and to build the SPARQL queries. Besides, as our method performs an end-to-end processing, our work is also related to several aspects presented in Sect. 2.3: identification of different types of SPARQL queries [19], detection of

[3] Biomedical question answering over interlinked data, `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=task2&q=4`

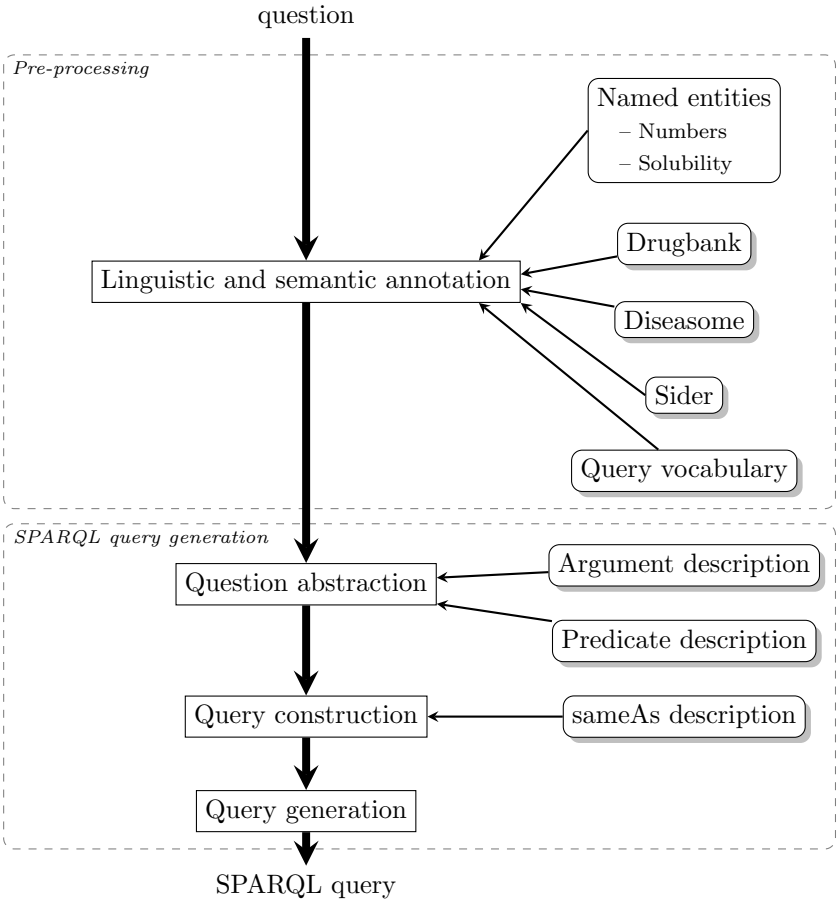

Fig. 1.: Global workflow of the system. Square boxes represent processing steps (they are detailed in Figures 2 to 7). Rounded boxes describe the resources used for the processing of questions and queries.

named entities [14], generation of SPARQL templates [13], etc.

The paper is structured as follows. We describe the proposed method in Sect. 4 and then the semantic resources available and developed for enriching the questions in Sect. 5. The evaluation of the method is presented in Sect. 6 and we finally discuss our results in Sect. 8.

4. Question translation into SPARQL Query

To translate natural language questions into SPARQL queries, we design a four-step rule-based approach, that relies on NLP methods, semantic resources and RDF triple description (see Fig. 1). Natural language questions are enriched with linguistic and semantic information (Sect. 4.1). This information is used for abstracting questions and for identifying relevant information (Sect. 4.2), and then for building the corresponding SPARQL query representations (Sect. 4.3) and for generating the SPARQL queries (Sect. 4.4).

We design our approach on 50 questions proposed by the task 2, *Biomedical question answering over interlinked data*, of the QALD-4 challenge, and evaluate it on 27 newly defined questions. A sample of the new test set is given at Sect. 6.1. We work with three KBs: Drugbank, Diseasome, and Sider described in Sect. 5. To illustrate the different tasks of our approach, we exemplify our approach on the several questions:

- *What is the side effects of drugs used for Tuberculosis?*
- *Which approved drugs interact with fibers?*
- *List drugs that lead to strokes and arthrosis.*
- *Give me drugs in the gaseous state.*

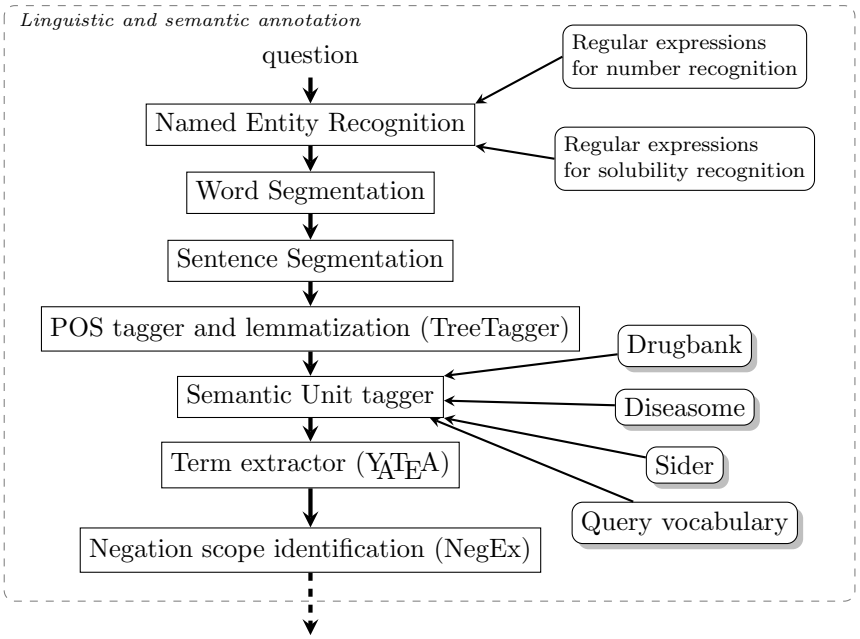

Fig. 2.: Linguistic and semantic annotation process. Square boxes represent the steps of linguistic and semantic analysis of questions. Rounded boxes indicate the resources used for the semantic annotation.

- *Which drugs have no side-effects?*
- *Which is the least common chromosome location?*
- *Which foods does allopurinol interact with?*

Note that the source questions are kept as provided despite the misspelling (first question for instance).

4.1. Linguistic and Semantic Annotation of Questions

The annotation step aims at associating a linguistic and semantic description with words and terms from the questions (see Fig. 2).

First, the linguistic annotation aims at parsing questions in order to identify numerical values (such as numbers and solubility values) and words. During this step, part-of-speech tags and lemmas are associated with words. To achieve that, we use the TreeTagger POS-tagger [28]. Figure 3 illustrates the obtained linguistic annotation of questions. POS tagging errors are intentionally kept in the examples (e.g. *List* tagged as noun instead of verb in 4c).

The objective of the semantic annotation is to identify semantic entities, i.e. terms together with the associated semantic types representing their meaning. Figure 4 displays the obtained semantic annotation of illustrated questions. This step relies on semantic resources, such as DrugBank, Sider and Diseasome (see Sect. 5), used in order to recognize semantic entities, such as disease names, side effects. The semantic entities recognition is based on the TermTagger Perl module[4]. For instance, in Figure 4a, *Tuberculosis* is recognized as an entity with two concurrent semantic types: `diseasome/disease/1154` and `sider/side-effects/C0041296`.

However, because semantic resources often suffer from low coverage [3,22], we also extract terms, which usually correspond to noun phrases relevant for the targeted domain, from the questions in order to improve the coverage of our approach. For instance, the terms *side effects of drugs* (Figure 4a) and *fibers* (Figure 4b) are extracted while none of them is provided by the semantic resources. The term extractor YaTeA[5] [2] is used for this task. It performs shallow parsing of the POS-tagged and lemmatized text by chunking it according to syn-

[4] `http://search.cpan.org/~thhamon/Alvis-TermTagger/`

[5] `http://search.cpan.org/~thhamon/Lingua-YaTeA/`

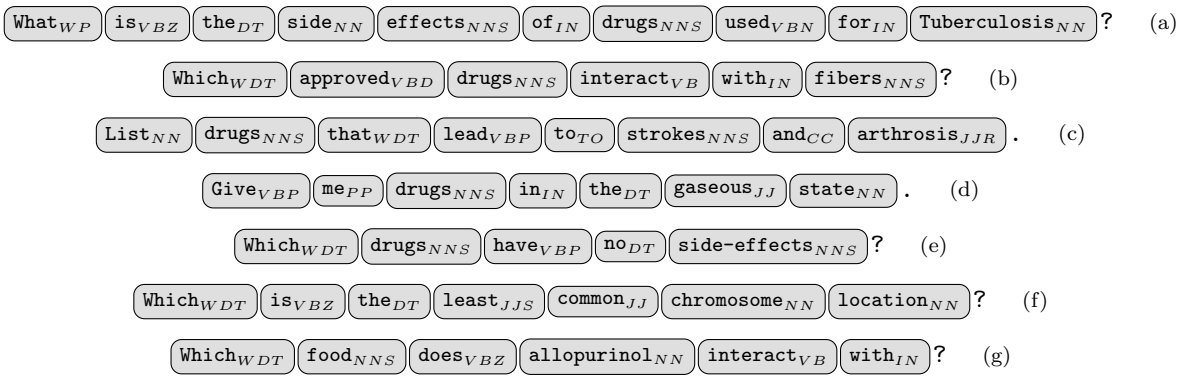

Fig. 3.: Examples of the linguistic annotation of questions issued from the QALD-4 challenge dataset. The source questions are kept as provided despite the misspellings they may contain (question (a) for instance). Gray rounded boxes represent words. Subscript text indicates Part-of-Speech tags computed by TreeTagger [28].

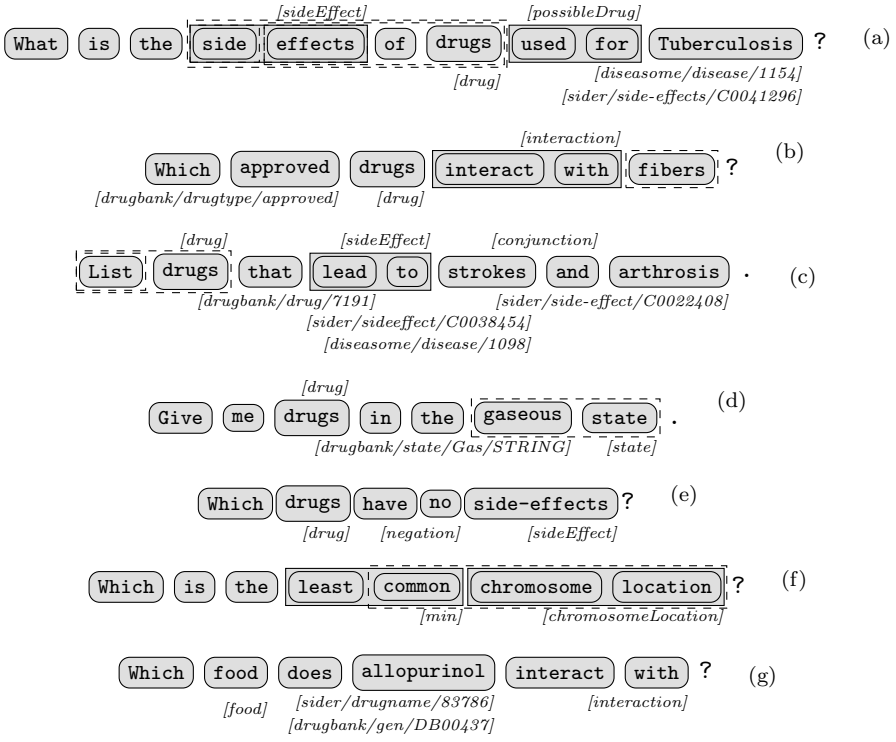

Fig. 4.: Examples of the question pre-processing. Gray rounded boxes represent words and semantic entities. Bracketed subscript texts are semantic types associated with semantic entities.

tactic frontiers (pronouns, conjugated verbs, typographic marks, etc.) in order to identify noun phrases. Then, parsing patterns are recursively applied and provide parsed terminological entities. These parsing patterns have been manually defined in a previous work [2]. They take into account the morpho-syntactic variation and represent basic syntactic dependencies within terminological entities. Each term is represented in a syntactic tree, while its sub-terms are also considered

as terms in the current configuration (e.g. *side effects drugs* gives *side effects* and *drugs* in Figure 4a). No semantic types are associated with the extracted terms.

The processing of questions also requires to identify expressions of negation (e.g. *no*) and quantification (e.g. *number of*, *least of*). Words expressing the negation (e.g. *no* in Figure 4e) are identified through regular expressions. Then, their scope is computed to detect terms that are negated within questions. For performing the task, we use the NegEx algorithm[6] [4]. We also collect and identify quantification expressions in the questions processed.

4.2. Question Abstraction

The question abstraction step aims at identifying relevant elements within questions and at building the representation of these elements (see Fig. 5). It relies on linguistic and semantic annotation associated with the question words detected at the previous step.

Before the identification of relevant elements, annotations are post-processed in order to disambiguate the generated semantic annotations. Indeed, annotated semantic entities may receive conflicting, concurrent or erroneous semantic types. For instance, in Figure 4g, *allopurinol* received two similar semantic types (one from DrugBank (**drugbank/gen/DB00437**) and one from Sider (**sider/drugname/83786**)), in Figure 4a, *Tuberculosis* is tagged as disease (**diseasome/disease/1154**) and side-effect (**sider/side-effects/C0041296**), while in Figure 4c, *lead* is erroneously tagged as drug (**drugbank/drug/7191**). The post-processing first aims at selecting those entities and semantic types that may be useful for the next steps. Therefore, semantic entities like *lead* is removed theyr are part of larger entities (*lead to*).

Besides, as part of this post-processing, larger terms which do not include other semantic entities are kept in order to increase the coverage of our approach. For instance, in Figure 4a, the terms *side effects of drugs* and *effects of drugs* are removed because two components (*side effects* and *drugs*) are semantically tagged, while in Figure 4b, the term *fibers* is kept.

Besides, in order to chose the correct predicate it may be necessary to consider semantic types in the context of words or phrases corresponding to the predicates. For instance, the phrase *interact with* is ambiguous and may correspond to two predicates:

- **foodInteraction** when its context contains semantic entity with the type **food** (Figure 4g),
- **InteractionDrug1** when its context contains semantic entity with the type **drug** (Figure 4b).

In that respect, we manually analyze the predicate names in order to define rewriting rules and to adjust (modification or deletion) semantic types associated with a given entity according to its context. Other rules may also modify or delete the entity itself. On the whole, we defined 44 contextual rewriting rules based on the vocabulary used in questions from the training set and on the documentation from the exploited KBs, mainly DrugBank[7].

In addition to the rewriting rules, supplementary disambiguation of the annotations is also performed during the *query construction* step when arguments of the predicate or the question topic share the same semantics and are connected. We define the question topic as the type of semantic entity which is the major context of the question [8]. For instance, in Figure 4a, the question topic is **sideEffects**.

For performing question abstraction, we identify information related to the query structure:

1. Definition of the Result form: negated terms and information related to coordination marks, aggregation operators, and requirements on specific result forms are recorded and will be used at the end of the *query construction* step or during the *query generation* step. Questions are scanned for identifying negated terms but also for identifying aggregation operation on the results, e.g. *number* for `count`, *mean* for `avg` or *higher* for `max`, and specific result forms such as boolean queries (`ASK`). Thus, in Figure 6f, the result form is the aggregation operation `min` applied on the object of the predicate **chromosomeLocation**, while the result form of other questions is `SELECT`. Also, presence of the

[6] `http://search.cpan.org/~osler/Lingua-NegEx/`

[7] `http://www.drugbank.ca/documentation`

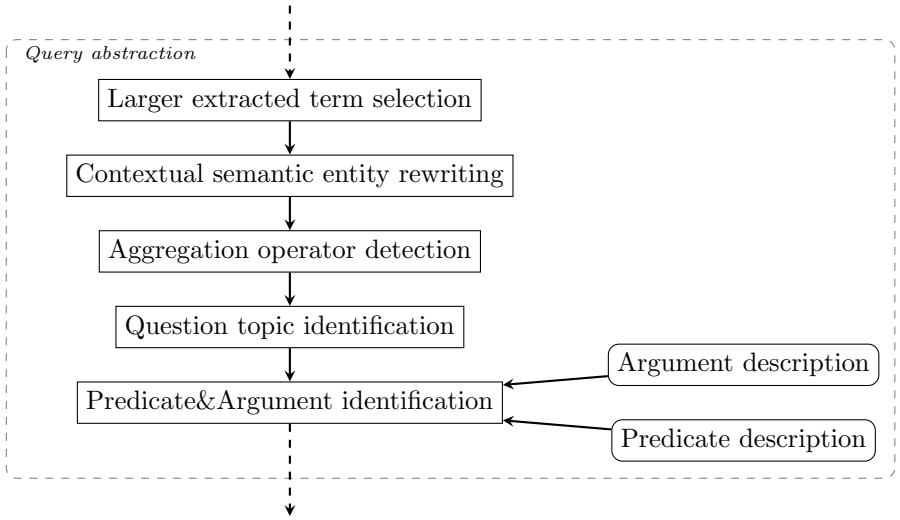

Fig. 5.: Question abstraction process. Square boxes represent the abstraction steps of questions. Rounded boxes indicate the resources used.

negated semantic entity *side-effects* in Figure 6e and of the coordination *and* in Figure 6c are recorded.

2. Identification of the Question topic: we assume that the first semantic entity with a given expected semantic type corresponds the question topic. The expected semantic types are those provided by the RDF subjects and objects issued from the resources. This information will be used during the *query construction* step. As illustration, the question topic is identified as **sideEffect** in Figure 6a and **chromosomeLocation** in Figure 6f.
3. Identification of Predicate and Arguments: we use linguistic representations of RDF schemas, i.e. frames which contain one predicate and at least two elements with associated semantic types. In that respect, potential predicates, subjects and objects of frames are identified among the semantic entities and then recorded in a table: entries are semantic types of the elements and refer to linguistic, semantic and SPARQL information associated with these elements. Subjects and objects are fully described in the table with the inflected and lemmatized forms of words or terms, the corresponding SPARQL types and indicators on their use as object or subject of a given predicate. Concerning the predicates, only the semantic types of their arguments are instantiated. Subjects and objects can be URI, RDF typed literals (numerical values or strings) and extracted terms (these are considered as elements of regular expressions).
   For instance, in Figure 6a, two predicates are identified: **sideEffect** and **possibleDrug**. Given the RDF schemas of these predicates, the expected arguments of the former are **sider/drugs** and **sider/side-effects**, while the arguments of the latter are **diseasome/diseases** and **drugbank/drugs**. Besides, in Figure 6d, the predicate *state* as well as the expected arguments **drugbank/drugs** and **Gas/String** are recognized.
4. Scope of coordination: arguments and predicate in the neighbourhood of coordination are identified. These elements are recorded as coordinated, e.g. the semantic entities *strokes* and *arthrosis* are related by the coordination *and* in Figure 6c.

Figure 6 presents graph representation and abstraction of the questions presented in Figure 4.

4.3. Query Construction

The objective of the *query construction* step is to associate previously identified elements and to build representation of the SPARQL graph pattern (introduced by the keyword `WHERE`). Figure 7

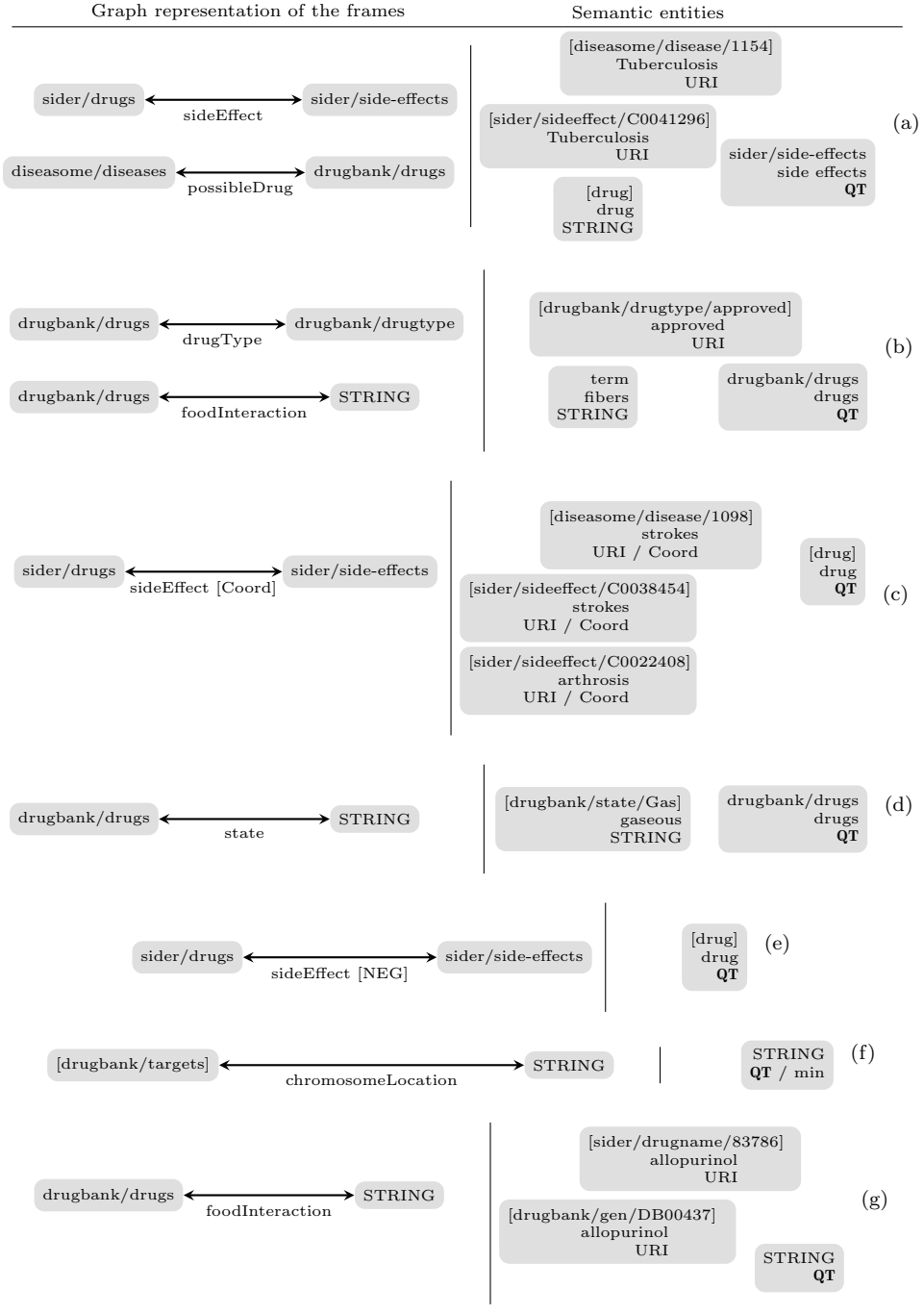

Fig. 6.: Examples of the question abstraction. The right part of the sub-figures displays graph representation of the identified frames: gray boxes represent subjects and objects of the predicates together with their semantic types, while edges represent predicates and their semantic types. The left part of the sub-figures represents semantic entities and terms identified in questions together with the associated information.

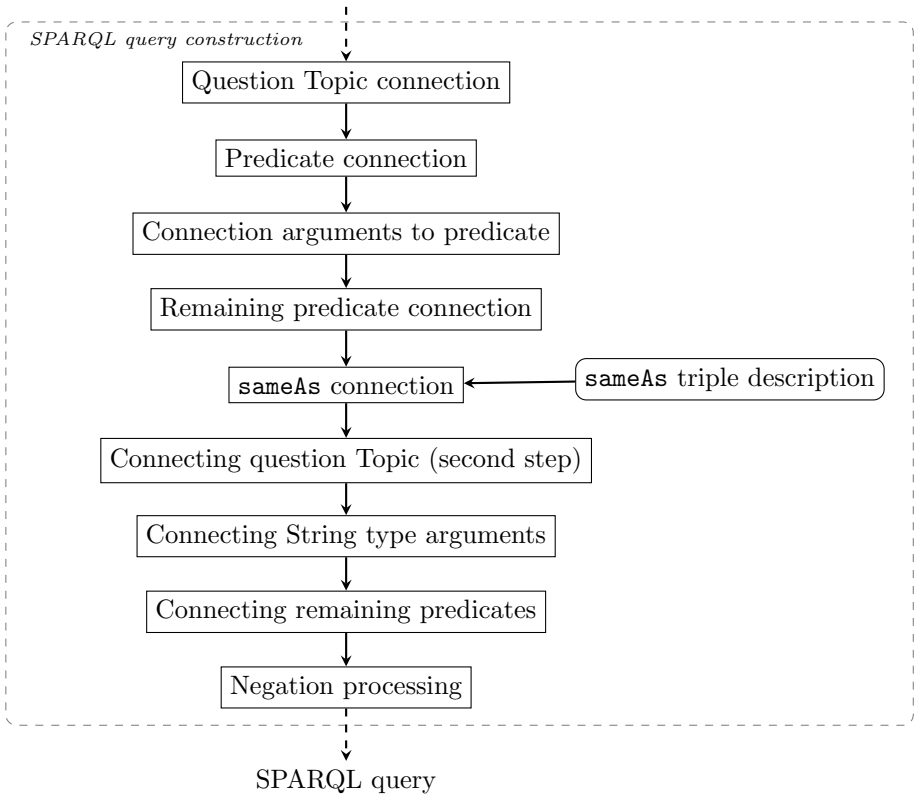

Fig. 7.: Query construction process. Square boxes represent construction steps of queries. Rounded boxes indicate the resource used.

presents the workflow of the query construction process and Figure 8 illustrates the construction of the queries corresponding to the examplified questions.

Thus, the predicate arguments are instantiated by URIs associated with the subjects, objects, variables, and numerical values or strings. For each question, we perform several associations:

1. The question topic is associated with one predicate argument and this is represented through a variable. Hence, this variable is associated with two elements: the question topic and one of the predicate arguments that matches the semantic type of this question topic. Notice that it is not necessary to associate all the predicate arguments that have the same semantic type with the question topic for now. Besides, at the end of this step, the question topic may remain non-associated with any predicate. In Figure 8a, the `?v0` variable represents the association between the question topic and the object (with the expected type **sider/side-effects**) of the **sideEffect** predicate. In Figure 8b, only the subject **drugbank/drugs** of the first **drugType** predicate is associated with the node corresponding to the question topic while the remaining association (subject **drugbank/drugs** of the **foodInteraction** predicate) will be processed during the next step. Performing this association at the beginning of the query construction is helpful for removing some ambiguities. For instance, in Figure 8a, the connection of the object of the **sideEffect** predicate prevents further use of the semantic type **sider/sideeffect/C0041296** of *Tuberculosis* (which was identified during the previous step, as illustrated in Figure 6a).
2. The predicate arguments are associated with semantic entities identified during the question abstraction: they concern elements referring to URIs. Moreover, each predicate with arguments in the coordination scope is dupli-

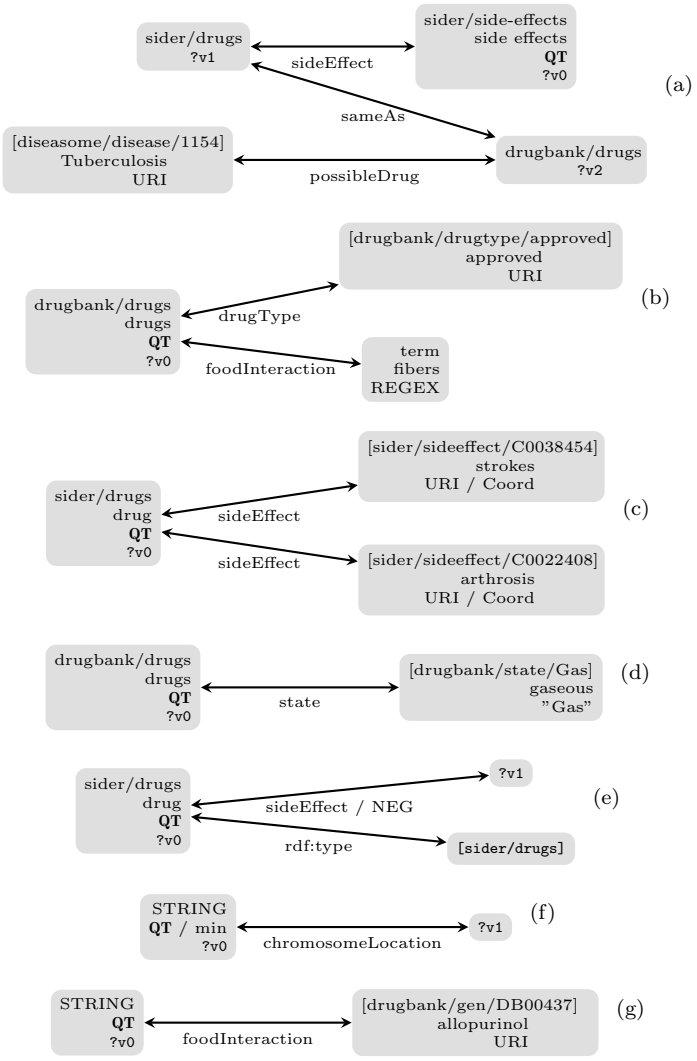

Fig. 8.: Examples of the query construction. Graphs represent the constructed queries and consequently, the SPARQL graph patterns. Gray boxes represent subjects and objects of the predicates instantiated by the semantic entities and terms. Variables associated to the predicate arguments are also displayed.

cated and arguments are also associated with semantic entities, if needed. Thus, in Figure 8a, *Tuberculosis* with the semantic type **diseasome/disease/1154** is associated with the subject of the **possibleDrug** predicate; in Figure 8b, **approved** is associated with the object of the **drugType** predicate; and in Figure 8d, the semantic entity *gaseous* is associated with the object of the **state** predicate. In Figure 8c, since the semantic entities *strokes* and *arthrosis* are coordinated, the **sideEffect** predicate is duplicated and these two semantic entities are associated with the predicate instances each. Note that, in this example, *strokes* is also implicitly disambiguated and not further considered as a disease.

3. The predicates are associated between them through their subjects and objects, and the association is then represented by a variable. For example, in Figure 8b, the subject of the **drugType** and **foodInteraction** predicates are associated. Hence, both are associated with the variable `?v0` which is already referred to as the question topic.

4. Predicates from different datasets are associated. We use the **sameAs** description to identify URIs referring to the same element. New variables are defined in order to associate two predicates. This kind of association occurs in the example in Figure 8a: the subject of the **sideEffect** predicate and the object of the **possibleDrug** predicate are associated through the **sameAs** predicate linking semantic entities **sider/drugs** and **drugbank/drugs**, respectively identified by the variables `?v1` and `?v2`.
5. The remaining question topics are associated with arguments of the **sameAs** predicate.
   The above examples do not require to perform such association.
6. The arguments corresponding to the `STRING` type are associated with the extracted terms. These arguments are related to the string matching operator `REGEX`. Thus, the terms are considered as string expressions. This is the case of the term *fibers* in Figure 8b which will be represented as regular expression in the next step.

At this point, the predicate arguments which remain unassociated are replaced by new variables in order to avoid empty literals. Finally, the negation operators are processed: predicates are marked as negated, while the arguments corresponding to negated terms are included in the new **rdf:type** predicate, if required. Thus, in Figure 8e, the object of the **sideEffect** predicate is negatively associated with the variable `?v1` and the **rdf:type** predicate with the object **sider/drugs** is added in the representation of the SPARQL graph pattern.

At this stage, each question is fully translated into a representation of the SPARQL query.

4.4. Query Generation

The SPARQL query representation built during the *query construction* step is used to generate the SPARQL query string. Figure 9 illustrates the generated queries which correspond to the examplified questions.

The query generation process is composed of two parts:

1. The generation of the result form which takes into account the expected type of the result form (e.g. `ASK` or `SELECT`), the presence of aggregation operators and the variable associated with the question topic. For all the examplified questions, the result form `SELECT`. Even if an aggregation operator is expected in Figure 9f, it requires the `GROUP BY` clause which will be processed during the next stage.
2. The generation of the graph pattern. This part consists of the generation of strings for representing each RDF triple and the filtering if the predicates are negated terms. This is the case of the examples from Figures 9a to 9e and Figure 9g. Also, in Figure 9d, the **state** predicate is replaced by the corresponding URI and its object is replaced by the string **Gas**. When aggregation operators are used, it is also necessary to recursively generate filters and sub-queries for computing the subsets of expressions, before their aggregation. Thus, in Figure 9f, two sub-queries are generated for counting the number of molecules per chromosome (given by the **chromosomeLocation** predicate) and the corresponding minimum number. Then, a filter is defined for the selection of the molecules per chromosome that have the minimum number.

The SPARQL queries are then submitted to a SPARQL end-point[8] and answers are collected for the evaluation.

5. Definition of the Semantic Resources

The method described above relies on: (1) the existing biomedical resources that provide information on semantic entities (Sect. 5.1); (2) additional resources specifically collected and built to support the method (Sect. 5.2).

5.1. Domain-specific Resources

To process the set of questions, we used the three following biomedical resources:

– DrugBank[9] KB is dedicated to drugs [32]. It merges chemical, pharmacological and phar-

[8] For our experiments, we use the SPARQL end-point provided by the QALD-4 challenge `http://vtentacle.techfak.uni-bielefeld.de:443/sparql`

[9] `http://www.drugbank.ca`

```
SELECT DISTINCT ?v0
WHERE {
    ?v1 <http://www4.wiwiss.fu-berlin.de/sider/resource/sider/sideEffect> ?v0.
    <http://www4.wiwiss.fu-berlin.de/diseasome/resource/diseases/1154>
    <http://www4.wiwiss.fu-berlin.de/diseasome/resource/diseasome/possibleDrug> ?v2.
    ?v1 <http://www.w3.org/2002/07/owl#sameAs> ?v2.
}
```

(a)

```
SELECT DISTINCT ?v0
WHERE {
   ?v0 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugType>
          <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugtype/approved>.
   ?v0 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/foodInteraction> ?v1.
   FILTER(REGEX(?v1,'fibers','i')).
}
```

(b)

```
SELECT DISTINCT ?v0
WHERE {
   ?v0 <http://www4.wiwiss.fu-berlin.de/sider/resource/sider/sideEffect>
         <http://www4.wiwiss.fu-berlin.de/sider/resource/side_effects/C0038454>.
   ?v0 <http://www4.wiwiss.fu-berlin.de/sider/resource/sider/sideEffect>
         <http://www4.wiwiss.fu-berlin.de/sider/resource/side_effects/C0022408>.
}
```

(c)

```
SELECT DISTINCT ?v0
WHERE {
   ?v0 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/state> "Gas".
}
```

(d)

```
SELECT DISTINCT ?v0
WHERE {
   FILTER NOT EXISTS {
      ?v0 <http://www4.wiwiss.fu-berlin.de/sider/resource/sider/sideEffect> ?v1.
   }
   ?v0 rdf:type <http://www4.wiwiss.fu-berlin.de/sider/resource/sider/drugs>.
}
```

(e)

```
SELECT DISTINCT ?v0
WHERE {
{
  SELECT DISTINCT (min(?v0count) as ?v0countmin)
  WHERE {
    ?v1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/chromosomeLocation> ?v0.
    {
      SELECT DISTINCT ?v0 (COUNT(?v0) as ?v0count)
      WHERE {
        ?v1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/chromosomeLocation> ?v0.
      }
      GROUP BY ?v0
    }
  }
}
{
   SELECT DISTINCT ?v0 (COUNT(?v0) as ?v0count)
   WHERE {
      ?v1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/chromosomeLocation> ?v0.
   }
   GROUP BY ?v0
}
FILTER (?v0countmin = ?v0count).
   ?v1 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/chromosomeLocation> ?v0.
}
```

(f)

```
SELECT DISTINCT ?v0
WHERE {
    <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00437>
        <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/foodInteraction> ?v0.
}
```

(g)

Fig. 9.: Examples of the query generation.

maceutical information from other available KBs. We exploit the documentation[10] of this resource to define the rewriting rules and regular expressions for the named entity recognition;
- Diseasome[11] is dedicated to diseases and genes linked among them by known disorder/gene associations [11]. It provides a single framework with all known phenotypes and disease gene associations, indicating the common genetic origin of many diseases. We exploit the RDF triples and the documentation of the resource to define the rewriting rules;
- Sider[12] is dedicated to adverse effects of drugs [16]. It contains information on marketed medicines and their recorded adverse drug reactions. Information is extracted from public documents and package inserts. Information available in Sider includes side effect frequency, drug and side effect classifications, as well as links to other data, such as drug-target relations. We use the documentation and the RDF triples of this KB.

The content of each resource is provided in a specific format: *subject predicate object* RDF triples. In that respect, we also exploit the RDF schemas of these resources to define the frames (see Sect. 5.2).

5.2. Additional Resources for the Question Annotation

On the basis of the RDF triples, frames are built from the RDF schemas in which the RDF predicate is the frame predicate, and subject and object of the RDF triples are the frame elements. This also includes the OWL `sameAs` triples. Several types of frame entities are isolated:

- As indicated, subject, object and predicate become semantic entities. At least one of them must occur in questions: in this way, the frames are the main resources for rewriting questions into queries;
- The vocabulary specific to questions is also built. It covers for instance the aggregation operators and the types of questions;
- RDF literals, issued from the named entity recognizer or the term extractor, complete the resources. The RDF literals are detected with specifically designed automata that may rely on the source KB documentation.

All these entities are associated with the expected semantic types which allow creating the queries and rewriting the RDF triples into SPARQL queries. In that respect, we can process several types of data (IRIs, strings, common datatypes or regular expressions) when literals are expected.

Most of the entities are considered and processed through their semantic types, although some ambiguous entities (e.g., interaction or class) are considered atomically. For these, the rewriting rules are applied contextually to generate the semantic entities corresponding to the frames (see Sect. 4.2). When using the queries, the semantic types become variables and are used for connecting the query edges.

6. Experiments and Results

6.1. Training and Test Question Set

50 questions from the training test gather training and test sets of the QALD-4 challenge. Separately, they show unbalanced complexity but taken together they provide a balanced training set. The evaluation is performed on 27 new questions. Questions from this new test set are similar to the QALD-4 questions but may differ as for the involved semantic entities or predicates. Our method is applied to this new test set without additional adaptations. Figure 10 presents a sample of questions from the new test set available at the following URL:
`http://perso.limsi.fr/hamon/Files/QALD/qald-4_biomedical_additional_test.xml`.

6.2. Evaluation Metrics

The generated SPARQL queries are evaluated through the answers they generate. The evaluation is performed with the following macro-measures [29]:

$$M_{\text{precision}} = \frac{\sum_{i=1}^{|q|} \frac{TP(q)}{TP(q)+FP(q)}}{|q|} \quad (1)$$

[10] `http://www.drugbank.ca/documentation`
[11] `http://diseasome.eu`
[12] `http://sideeffects.embl.de`

Which foods does fluvoxamine interact with?
Are there drugs that target the Probable arabinosyltransferase A?
Which genes are associated with subtypes of rheumatoid arthritis?
Which disease has the highest degree?
Which targets are involved in immune function?

Fig. 10.: Example of natural language questions from the new test set.

$$M_{\text{recall}} = \frac{\sum_{i=1}^{|q|} \frac{TP(q)}{TP(q)+FN(q)}}{|q|} \quad (2)$$

$$M_{\text{F-measure}} = \frac{2 \times M_{\text{precision}} \times M_{\text{recall}}}{M_{\text{precision}} + M_{\text{recall}}} \quad (3)$$

where $TP(q)$ are the correct answers, $FP(q)$ are the wrong answers and $FN(q)$ are the missing answers for the question q.

Through the use of macro-measures, we equally consider all the questions independently on the number of expected answers for a given SPARQL query.

6.3. Global Results

Table 1 presents the overall results obtained on the training and test sets. On the test set, the macro-F-measure is 0.78 with 0.81 precision and 0.76 recall, while on the training set, the macro-F-measure is 0.86 with 0.84 precision and 0.87 recall.

Query set	Training (50 Q)	Test (27 Q)
Correct Queries	39	20
M-precision	0.84	0.81
M-recall	0.87	0.76
M-F-measure	0.86	0.78

Table 1: Results obtained with the training and test sets.

Our method always proposes syntactically correct SPARQL queries for all natural language questions. On the test set, concerning the answers generated over linked data, 20 questions provide the exact expected answers, two questions return partial answers, and five questions return erroneous answers. On the training set, 39 SPARQL queries (out of the 50 questions) provide the expected answers, six questions return partial answers, and five questions return no answers.

6.4. System Performance

We analyzed the system performance when translating 50 questions from the training set, on a computer with 4 Gb of memory and a 2.7GHz dual-core CPU. Figure 11 presents the running time for each query according to the pre-processing sub-steps (named entity recognition, word and sentence segmentation, POS tagging, semantic entity tagging, term extraction with YaTeA and negation scope identification with NegEx) and the question translation into a SPARQL query (Question2SPARQLQuery). Most of the processing time is dedicated to the TermTagger which aims at recognizing the semantic entities. With the internal Ogmios processing (i.e., mainly the control of inputs and outputs), each question is processed in 2 seconds on the average.

Figure 12 shows the overall system performance according to the number of questions to be processed. The variation of running time when processing one question and the whole set of questions is less than two seconds on the training set.

7. Discussion

7.1. Findings

Overall, our approach exhibits good results, with 0.78 F-measure on the newly created test set. This value is lower than the F-measure obtained on the training set because no additional rules or adjustments were made for processing the questions from the new set. It is noteworthy that this value would have been higher if the `sameAs` predicate had been correctly described as symmetric. Indeed, we noticed that in the reference data, for three questions, the generated SPARQL query was correct but the SPARQL end-point did not return the expected answers. By switching the arguments of the `sameAs` predicate in the queries, we observed

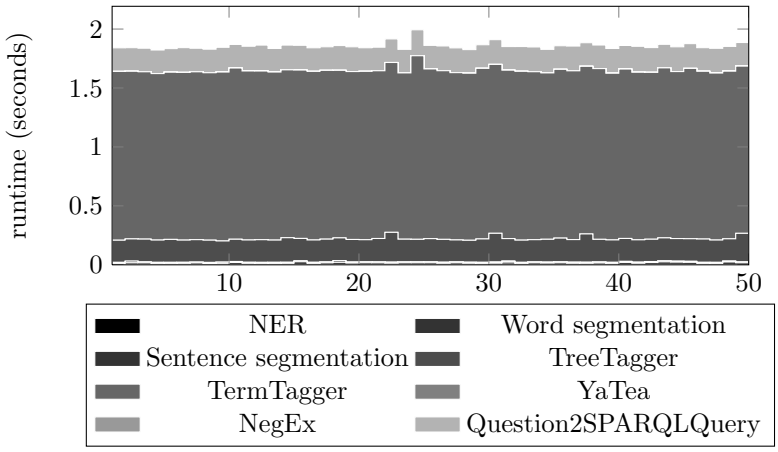

Fig. 11.: Performance per sub-step for 50 questions from the training set.

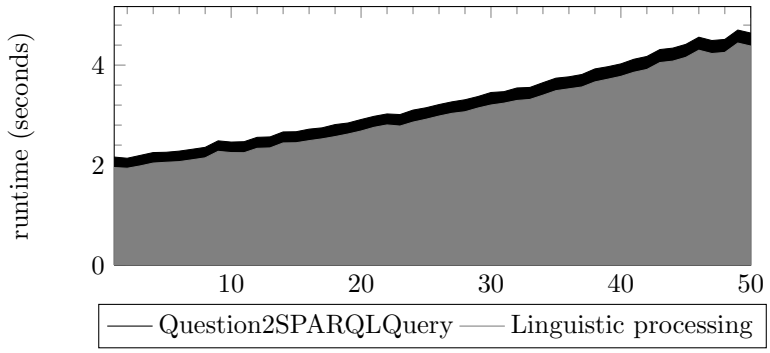

Fig. 12.: System performance for an increasing number of questions (1 to 50) from the training set.

that the expected answers were returned then. Although the `sameAs` predicate is symmetric by definition, it actually appears that the instances of this predicate do not encode the expected reflexivity of this relation in the source KBs.

7.2. Comparison with Existing Works

We propose two ways for comparing our work with the existing ones: either the methods or the reference data are comparable. Hence, we can compare our work with those end-to-end works presented in Section 2.2. In Table 2, we indicate the available evaluation numbers for Precision, Recall and F-measure. We can observe that our system provides competitive results.

We can also compare our system with those that participated in the QALD-4 challenge. In Table 3, we indicate the official results of the challenge: our system rates second among the three participating systems. The first system exploits Grammatical Framework grammar based on formal syntax, while the third one proposes a semi-manual approach combining automatic POS-tagging and manual transformation of questions into queries. The results provided by our system are close to the best system of the challenge.

On the whole, we can observe that the proposed method is competitive by comparison with existing work, and is also portable to new datasets.

Similar works applied on general-language datasets show less impressive results. For instance, comparable approach (linguistic annotation, syntactic analysis and scoring of the right answers) applied to comparable material (QALD-3 DBpedia) reaches 0.32 F-measure [7]. We assume that processing of data from specialized areas, for which terminologies and semantic resources are available, provide the possibility to descrive the involved concepts and scenarii with more detail. As

System	Ref. data	P	R	F
(Unger et al., 2012) [31]	QALD-2	0.61	0.63	0.62
(Ben Abacha et al., 2012) [1]	Med. Journals	0.62		
(Lukovnikov et al., 2014) [20]	QALD-3			0.4-0.8
Our system	QALD-4	0.81	0.76	0.78

Table 2: Evaluation results of comparable end-to-end systems.

System	Approach	P	R	F
GFMed [21]	GF grammar	0.99	1	0.99
Our system	NLP, sem. resources	0.87	0.82	0.85
RO_FII	semi-manual	0.16	0.16	0.16

Table 3: Evaluation results from the QALD-4 challenge.

result, performance of automatic systems can be higher there.

7.3. Error Analysis

We performed an analysis of erroneous or partial answers, other than those caused by `sameAs`. The analysis shows that most of the errors are due to the management of ambiguities in the questions, which has also been noticed in a previous work [20]. These errors are mainly related to:

- The annotation of semantic entities. For instance, in *Which genes are associated with breast cancer?*, *breast cancer* is correctly annotated, while the reference assumes mistakenly that it should concern the semantic entity *Breast cancer-1*.
- The expected meaning of the terms in the questions. Semantic entities mentioned in some questions may refer to specific entities while in other questions they may refer to the general entities. For instance, in *What are enzymes of drugs used for anemia?*, the semantic entity *anemia* refers to all types of anemia (*Hypercholanemia*, *Hemolytic anemia*, *Aplastic anemia*, etc.), and not specifically to elements that contain the label *anemia*.

These two main problems can be solved by using regular expressions in SPARQL graphs rather than URIs. However, we must test the influence of this modification on each query individually.

Other erroneous answers happen during the question abstraction step when the question topics are wrongly identified or when the contextual rewriting rules are not applied. Errors may also occur during the query construction step: the method may abusively connect predicate arguments and semantic entities or, on contrary, it may not consider all the identified semantic entities. Further investigations have to be carried out to solve these limitations.

Besides, during the design of queries, we had difficulties to express some constraints in SPARQL. For instance, the question *Which approved drugs interact with calcium supplements?* requires to define regular expression with the term *calcium supplement* while this term is only mentioned in coordination with other supplements in the exploited KBs (e.g. *Do not take calcium, aluminum, magnesium or Iron supplements within 2 hours of taking this medication.*). We assume that solving this difficulty requires a more sophisticated NLP processing of the textual elements of the RDF triples, such as parsing of the RDF textual elements, named entity and term recognition, identification of discontinuous terms and term variants.

7.4. Reproducibility of our Method

Our method is fully automated, once the rewriting rules have been defined. A key issue related to the reproducibility concerns the evolution of KBs. In case they are updated, it is only required to rebuild the semantic resources used for identifying the semantic entities. Yet, for managing the change of the structure of the KBs, entire frames must be regenerated. This is one direction of our ongoing research work. Moreover, the addition of

new resources such as Dailymed[13], is also related to these two problems.

8. Conclusion

We proposed a rule-based method to translate natural language questions into SPARQL queries. The method relies on linguistic and semantic annotation of questions with NLP methods, semantic resources and RDF triples description. We designed our approach on 50 biomedical questions proposed by the QALD-4 challenge, and tested it on 27 newly created questions. The method achieves good performance with 0.78 F-measure on the set of 27 questions.

Further work aims at addressing the limitations of our current method including the management of term ambiguity, the question abstraction, and the query construction. Moreover, to avoid the manual definition of the dedicated resources required by our approach (frames, specific vocabulary and rewriting rules), we plan to investigate how to automatically build these dedicated resources from the RDF schemas of the Linked Data set. This perspective will also facilitate the integration of other biomedical resources such as Dailymed or RxNorm [27], and the use of our method in text mining applications.

The method for translating questions into SPARQL queries is implemented as a Perl module and is available at `http://search.cpan.org/~thhamon/RDF-NLP-SPARQLQuery/`. The new set of 27 questions is also available at the following URL:
`http://perso.limsi.fr/hamon/Files/QALD/qald-4_biomedical_additional_test_withanswers.xml`.

Acknowledgments

This work was partly funded through the project POMELO (*PathOlogies, MEdicaments, aLimentatiOn*) funded by the MESHS (Maison Européenne des Sciences de l'Homme et de la Société) under the framework *Projets Émergents*. We thank Arthur Plesak for his editorial assistance. We are thankful to the reviewers for their useful comments and advices which permitted to improve the quality of the manuscript.

[13] `http://dailymed.nlm.nih.gov/`

References

[1] A. B. Abacha and P. Zweigenbaum. Medical question answering: Translating medical questions into SPARQL queries. In *ACM SIGHIT International Health Informatics Symposium (IHI 2012)*, 2012.

[2] S. Aubin and T. Hamon. Improving term extraction with terminological resources. In T. Salakoski, F. Ginter, S. Pyysalo, and T. Pahikkala, editors, *Advances in Natural Language Processing (5th International Conference on NLP, FinTAL 2006)*, number 4139 in LNAI, pages 380–387. Springer, August 2006.

[3] O. Bodenreider, T. C. Rindflesch, and A. Burgun. Unsupervised, corpus-based method for extending a biomedical terminology. In *Workshop on Natural Language Processing in the Biomedical Domain (ACL2002)*, pages 53–60, 2002.

[4] W. Chapman, W. Bridewell, P. Hanbury, G. Cooper, and B. Buchanan. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform. 2001 Oct;34(5):*, 34(5):301–10, 2001.

[5] D. Damljanovic, M. Agatonovic, and H. Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I*, ESWC'10, pages 106–120, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] M. Damova, D. Dannélls, and R. Enache. Multilingual retrieval interface for structured data on the web. In *Workshop on Natural Language Interfaces for Web of Data*, 2014.

[7] C. Dima. Intui2: A prototype system for question answering over linked data. In *CLEF QALD-3*, pages 1–12, 2013.

[8] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. In *Proceedings of ACL-08: HLT*, pages 156–164, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[9] E. Franconi, C. Gardent, X. Juarez-Castro, and L. Perez-Beltrachini. Quelo natural language interface: Generating queries and answer descriptions. In *Workshop on Natural Language Interfaces for Web of Data*, 2014.

[10] A. Freitas, J. ao C. Pereira da Silva, and E. Curry. On the semantic mapping of schema-agnostic queries: A preliminary study. In *Workshop on Natural Language Interfaces for Web of Data*, 2014.

[11] V. Janjić and N. Pržulj. The core Diseasome. *Mol Biosyst*, 8(10):2614–2625, Aug 2012.

[12] E. Kaufmann and A. Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *Proceedings of the Forth European Semantic*

Web Conference (ESWC 2007), Innsbruck, Austria, June 2007.

[13] J.-D. Kim and K. Cohen. Triple pattern variation operations for flexible graph search. In *Workshop on Natural Language Interfaces for Web of Data*, 2014.

[14] Y. Kim, Y. Hahm, D. Hwang, and K.-S. Choi. A non-morphological entity boundary detection approach for Korean text. In *Workshop on Natural Language Interfaces for Web of Data*, 2014.

[15] N. Kuchmann-Beauger and M.-A. Aufaure. Natural language interfaces for DataWarehouses. In *8èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2012), Bordeaux*, volume B-8 of *RNTI*, pages 83–92, Paris, Juin 2012. Hermann.

[16] M. Kuhn, M. Campillos, I. Letunic, L. J. Jensen, and P. Bork. A side effect resource to capture phenotypic effects of drugs. *Molecular Systems Biology*, 6(1), 2010.

[17] V. le Clément de Saint-Marcq, Y. Deville, C. Solnon, and P.-A. Champin. Un solveur léger efficace pour interroger le web sémantique. In *JFPC 2012*, 2012.

[18] J. Lehmann and L. Bühmann. AutoSPARQL: Let users query your knowledge base. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications*, volume Part I, pages 63–79, 2011.

[19] L. Letouzey and A. Gabillon. Implementation d'un modèle de contrôle d'accès pour les documents RDF. In *SarSsi 2010*, 2010.

[20] D. Lukovnikov and A.-C. Ngonga Ngomo. SESSA - keyword-based entity search through coloured spreading activation. In *Workshop on Natural Language Interfaces for Web of Data*, 2014.

[21] A. Marginean. GFMed: Question answering over biomedical linked data with grammatical framework. In *CLEF 2014 Working Notes Papers*, 2014.

[22] A. T. McCray, A. C. Browne, and O. Bodenreider. The lexical properties of the Gene Ontology (GO). In *Proceedings of the AMIA 2002 Annual Symposium*, pages 504–508, 2002.

[23] G. Montoya, M.-E. Vidal, and M. Acosta. A heuristic-based approach for planning federated SPARQL queries. In *COLD*, 2012.

[24] C. Pradel, O. Haemmerlé, and N. Hernandez. Expression de requêtes SPARQL à partir de patrons : prise en compte des relations (regular paper). In A. Mille, editor, *Journées Francophones d'Ingénierie des Connaissances (IC), Chambéry, 16/05/2011-20/05/2011*, pages 771–786. Presses Universitaires des Antilles et de la Guyane, mai 2011.

[25] C. Pradel, O. Haemmerlé, and N. Hernandez. Des patrons modulaires de requêtes SPARQL dans le système SWIP. In *Journées Francophones d'Ingénierie des Connaissances (IC)*, pages 412–428, juin 2012.

[26] A. Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011.

[27] RxNorm, a standardized nomenclature for clinical drugs. Technical report, National Library of Medicine, Bethesda, Maryland, 2009. Available at `www.nlm.nih.gov/research/umls/rxnorm/docs/index.html`.

[28] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In D. Jones and H. Somers, editors, *New Methods in Language Processing Studies in Computational Linguistics*, 1997.

[29] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[30] V. Tablan, D. Damljanovic, and K. Bontcheva. A natural language query interface to structured information. In L. N. in Computer Science Volume 5021, editor, *The Semantic Web: Research and Applications*, pages 361–375, 2008.

[31] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *WWW*, pages 639–648, 2012.

[32] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey. Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Research*, 34:D668–D672, 2006. Database issue.