

QA³: a Natural Language Approach to Statistical Question Answering

Editor(s): Stamatia Dasiopoulou, Pompeu Fabra University, Spain; Georgios Meditskos, Centre for Research and Technology Hellas, Greece; Leo Wanner, ICREA and Pompeu Fabra University, Spain; Stefanos Vrochidis, Centre for Research and Technology Hellas, Greece; Philipp Cimiano, Bielefeld University, Germany

Maurizio Atzori ^a, Giuseppe M. Mazzeo ^b and Carlo Zaniolo ^b

^a *Department of Maths and Computer Science, University of Cagliari, V. Ospedale 72, 09124 Cagliari, Italy*
E-mail: atzori@unica.it

^b *Computer Science Department, UCLA, 3532D Boelter Hall Los Angeles, CA 90095-1596, CA, USA*
E-mail: mazzeo@cs.ucla.edu, zaniolo@cs.ucla.edu

Abstract. In this paper we present QA³, a question answering (QA) system over RDF cubes. The system first tags chunks of text with elements of the knowledge base, and then leverages the well-defined structure of data cubes to create the SPARQL query from the tags. For each class of questions with the same structure a SPARQL template is defined, to be filled in with SPARQL fragments obtained by the interpretation of the question. The correct template is chosen by using an original set of regex-like patterns, based on both syntactical and semantic features of the tokens extracted from the question. Preliminary results obtained using a very limited set of templates are encouraging and suggest a number of improvements. QA³ can currently provide a correct answer to 27 of the 50 questions of the test set of the task 3 of QALD-6 challenge, remarkably improving the state of the art in natural language question answering over data cubes.

Keywords: question answering, RDF data cube, statistical queries, natural language,

1. Introduction

Governments of several countries have recently begun to publish information about the public expenses, using the RDF data model [4] in order to improve transparency. The need to publish statistical data, which concerns not only Governments but also many other organizations, has pushed the definition of a specific RDF-based model, i.e., the RDF data cube model [5], whose first draft was proposed by W3C in 2012, and its current version was published in January 2014. The availability of data of public interest from different sources has fostered the creation of projects, such as LinkedSpending [1], that collect statistical data from several organizations, making them available as an RDF knowledge base, following the Linked Data principles. However, while RDF data can be efficiently queried using the powerful SPARQL language,

only technical users can effectively extract human-understandable information. The problem of providing a user-friendly interface that enables non-technical users to query RDF knowledge bases has been widely investigated during the last few years. All the proposed solutions provide some systems which translate “easy” operations made by the user into a SPARQL query, whose result represents the answer to the intention that the user expressed through its operations. More specifically, in the following some of the existing approaches are listed.

Exploratory browsing allows users to navigate through the triples of the RDF graph by starting from an entity (node) and then clicking on a property (edge), thus moving to another entity. Although the users do not need to know beforehand the exact names of properties, this approach can be effectively used only for exploring the graph in the proximity of the initial entity.

Furthermore, this approach is not suitable for aggregate queries, which are the real first-class citizens in the world of RDF data cubes.

Faceted search supports a top-down search of RDF graph, by starting with the whole dataset as potential results, and enabling the user to progressively restrict the results by defining some constraints on the properties of the current result set [12]. This approach was recently applied to the RDF data cubes [21], following the long tradition of graphical user interfaces for OLAP analysis, based on charts representing different kind of aggregations of the underlying data.

Another user-friendly system for querying RDF data is SWiPE [7], which makes the infobox of Wikipedia pages editable. The user can type the constraints using the value fields of the infobox, according to the *By-Example Structured Query* (BESTQ) paradigm. While this paradigm is very effective for finding list of entities with specific properties starting from a wikipage, its generalization to the RDF data cubes is not trivial.

Natural language interfaces are the most challenging solutions to the problem of question answering over RDF data. These systems let the user type any question in natural language and translate it into a SPARQL query. The current state-of-the-art system is Xser [23], which was able to yield an F-score equal to 0.72 and 0.63 in the, respectively, 2014 and 2015 QALD challenges [2]. Although its accuracy is far from being very high, Xser largely won over the other participating systems. This witnesses the fact that translating natural language questions into SPARQL queries is a really hard task.

The difficulty of the task can be reduced by using a *controlled natural language* (CNL), i.e., a language which is generated by a restricted grammar and can be efficiently interpreted by machines, yet natural enough to be easily understood by non-technical users. Some of the CNL-based systems that participated in past QALD challenges are Squall2sparql [10] and GFmed [20]. While the accuracy of both systems is very high, the former suffers from usability issues, since the user needs to type non-natural phrases in the question, while the latter has scalability problems, since the number of rules of its grammar depends on the size of the knowledge base.

Authors in [22] also proposed CANaLI, a system that overcomes the above issues, yet providing high accuracy. CANaLI relieves the usability issues derived from the use of a CNL by means of an interactive interface having no scalability issues, since it combines a restricted set of rules, which do not depend on the

knowledge base, with the use of an index, queried at run-time to find the tokens that are consistent with the grammar of the CNL and the semantics of the knowledge base.

None of the previously proposed systems, either based on full or controlled natural language, has been specialized for question answering over RDF data cubes. Question answering over RDF cubes is, indeed, a brand new challenge, which raises issues different from those of question answering on a “general” knowledge base [14]. In fact, questions in this context are likely to be very specific, i.e., oriented towards extracting statistical information. Thus, a system called to interpret these questions must be as much accurate as possible in interpreting specific features of the typical OLAP queries, such as different kind of aggregations or constraints on dimensions. Furthermore, questions on statistical data are more sensitive to misinterpretations. For instance, while a partial interpretation of a general question might also yield an acceptable answer (maybe with non perfect precision), in an aggregation query missing a constraint is likely to yield a totally wrong answer.

In this paper we propose QA³ (pronounced as *QA cube*), a question answering system tailored for RDF cubes. The paper is organized as follows. First, in Section 2 the model used to represent data cubes in RDF is described. Then, the approach used by QA³ is presented in Section 3, and preliminary experimental results are reported in Section 4. Section 5 reviews the related work in the area of Statistical Question Answering over RDF data. A discussion on the specific problem of answering OLAP-related questions through natural languages and lesson learnt in this work, including possible improvements, are presented in Section 6. Finally, Section 7 concludes the paper, describing the ongoing work.

2. The RDF data cube model

The RDF Data Cube Vocabulary [5] has been recently proposed as a W3C standard for representing multi-dimensional statistical data using the RDF format. The model underlying this vocabulary is based on the SDMX model [3], an ISO standard for representing statistical data that can be shared among organizations.

In the following, we briefly describe the RDF data cube model, focusing our attention on the features depicted in Fig. 1, that are the most relevant for the

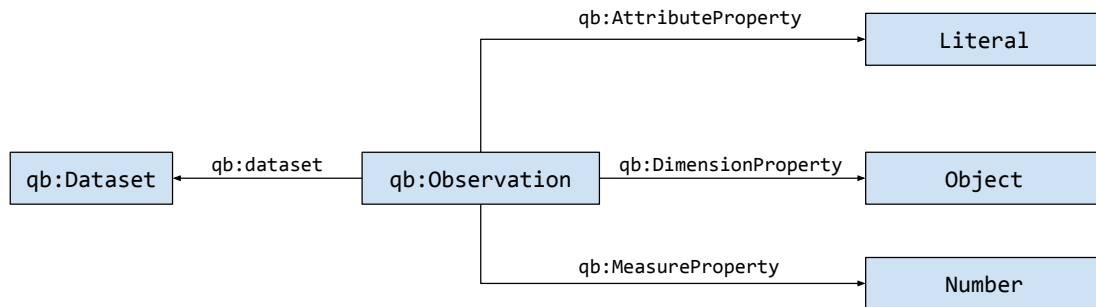


Fig. 1. The RDF data cube model simplified

question answering task. A data cube, called *dataset*, is an instance of the class `qb:Dataset`¹. The cells of the data cube, called *observations*, are instances of the class `qb:Observation`. Each observation can be associated with one or more *measures*, *attributes*, and *dimensions*. The properties that define measures are of type `qb:MeasureProperty`. These properties associate the observations with quantitative values (e.g., an amount of money). Attributes and dimensions are of type `qb:AttributeProperty` and `qb:DimensionProperty`, respectively. They both define the context of the observation, but the former associates the observations with literal values (e.g., the year), while the latter associates the observations with objects (e.g., the department).

Fig. 2 depicts a set of triples that use the RDF data cube dictionary. Assuming that the prefix `ex` denote an example namespace, the first four triples define the metadata of the dataset, in particular: (i) the dataset object with URI `ex:expenses`, (ii) a dimension with URI `ex:department`, (iii) an attribute with URI `ex:refYear`, and (iv) a measure with URI `ex:amount`. The fifth triple defines an observation with URI `ex:obs1`, and the sixth triple states that this observation belongs to the `ex:expenses` dataset. The last three triples specify the department (`ex:PublicWorks`), the year (2015), and the amount (5432.1) of the observation `ex:obs1`. In a typical RDF cube several observations would be defined (here, we limit our example to one observation for sake of brevity), each associated with the measures, dimensions, and attributes defined by the metadata. Typical queries on such data combine aggregation with simple constraints defined on the attributes/measures of the observation. For instance, a user might be inter-

ested in the total amount spent by the department of Public Works in 2015, which can be found using the following SPARQL query:

```
SELECT sum(xsd:decimal(?measure)) {
  ?observation qb:dataset ex:expenses.
  ?observation ex:amount ?measure.
  ?observation ex:department
    ex:PublicWorks.
  ?observation ex:refYear
    "2015"^^xsd:gYear.
}
```

3. An overview of QA³

QA³ is a system that translates questions posed in natural language into SPARQL queries, assuming that the answer to the questions can be provided using the knowledge base (KB) “known” by the system. In particular, the KB contains a set of data cubes stored using the RDF data cube model described in Section 2. QA³ works in three steps:

- the question is tagged with elements of the KB, belonging to the same dataset. This step also detects which dataset is the most appropriate to answer the question;
- the question is tokenized, using Stanford tokenizer and POS tagger [19], and the tags obtained at this step are augmented using the tags obtained at the previous step. The sequence of tokens is then matched against some regular expressions, each associated with a SPARQL template;
- the chosen template is filled with the actual clauses (constraints, filters, etc.) by using the tags and the structure of dataset.

In the following, each step is described in more detail.

¹qb is the prefix for the namespace <http://purl.org/linked-data/cube#>

ex:expenses	rdf:type	qb:Dataset
ex:department	rdf:type	qb:DimensionProperty
ex:refYear	rdf:type	qb:AttributeProperty
ex:amount	rdf:type	qb:MeasureProperty
ex:obs1	rdf:type	qb:Observation
ex:obs1	qb:dataset	ex:expenses
ex:obs1	ex:department	ex:PublicWorks
ex:obs1	ex:refYear	"2015"^^xsd:gYear
ex:obs1	ex:amount	"5432.1"^^xsd:double

Fig. 2. Example of of possible triples using the RDF data cube dictionary

3.1. Tagging the questions with elements of the KB

Given a string Q representing a question, for each dataset D we try to match the literals in D against chunks of Q . Before performing the matching, the strings are normalized, by removing the stop words and performing lemmatization. The result of the matching between Q and the dataset D can be represented as set of pairs $\langle C, T \rangle$, where C is a chunk of Q and T is a set of triples in D . Each matching is associated with a quality measure (or score), which is based on the weighted percentage of Q that is covered by tagged chunks. Chunks have weight based on the apriori probability of being tagged and their length. This quality measure enables the system to choose the dataset which most likely has to be used to provide an answer to the question. Depending on the setting (e.g., in order to improve recall), the top- k datasets can be used as candidates, and the final choice is made in the following steps performed by the system.

3.2. Finding the SPARQL query template

The domain in which QA³ operates is quite structured, especially if we compare it to that in which general purpose question answering systems are required to work. As a consequence, the meaningful questions (i.e., questions that can be answered using the available KB) are likely to have a SPARQL translation which follow a limited set of well defined *templates*. For instance, if the user wants to know how much his city spent for public works in 2015, the question has to contain all the elements needed to detect the dataset to be used, the measure to aggregate and the aggregation function, and the constraints to be applied to restrict the computation on the observations in which the user is interested. This question, like a wide range of similar questions, can be answered by using the following

SPARQL query template:

```
SELECT sum(?measure) {
  ?observation qb:dataset <dataset> .
  ?observation <measure> ?measure .
  <constraints>
}
```

where $\langle \text{dataset} \rangle$ and $\langle \text{measure} \rangle$ must be replaced with the correct URIs, and $\langle \text{constraints} \rangle$ must be replaced with a set of triples specifying the constraints for the variable $?observation$, representing the observations. These constraints are extracted from the question and represented as *SPARQL fragments*, replacing the $\langle \text{constraints} \rangle$ placeholder in the main template.

Another possibility is asking which department has spent the most in 2015. In this case, the SPARQL query will select the value of the dimension that represents the department, and will provide the one associated with the largest sum of spent amount of money. Even this question, like most of the questions that can be asked in this domain, can be answered using a SPARQL query which has a structure that can be used to answer many similar questions.

In order to leverage the typical homogeneity of the structures of these questions, we implemented a system which allows to define a set of SPARQL templates and to automatically detect the one that should be used to provide an answer. To this end, each template is associated with one (or possibly more) regular expressions built on the tokens of the questions. The tokens are obtained using the Stanford parser [19], which tokenizes the question and annotates each token with its lemma, its POS (part of speech) tag, and its NER (named entity recognition) tag. We augment the annotations with the elements of the knowledge base (dataset, measure, dimension, attribute, literal) provided by the previous step, and with a tag repre-

sending a possible aggregate function. For the latter, a specific (very small) dictionary has been defined, using the words that are used most often to name aggregation functions (e.g., sum, total, average, maximum, etc.).

Thus, for each token we have the following features:

1. the POS tag
2. the lemma
3. the word (i.e., the original text)
4. the NER tag
5. the KB tag (S: dataset, M: measure, D: dimension, A: attribute, E: entity, L: literal, O: none)
6. the aggregation tag (S: sum, A: average, M: max, N: min, O: none)

A *generalized-token* is defined as a 6-tuple, where the *i*-th element represents the possible set of values for the *i*-th feature of the actual tokens (the features are assumed to follow the order in which they are listed above). For instance, the generalized-token $WP|WDT, _, _, _, !E$ matches against actual tokens that have WP or WDT as POS tag, any lemma, word, and NER tag ($_$), and the token must not be annotated as entity (!E). The features that are not explicitly specified in the generalized-token (in this case the aggregation tag), are allowed to take any value (i.e., $_$ is implicitly assumed).

A generalized-token can be followed by a # symbol and by a string that represents a label name, that can be used to get the actual token/s that matched the generalized-token. The generalized-tokens can be used to build more complex regular expressions. Fig. 3 reports the 7 templates currently used by QA³, with the corresponding regular expression.

We describe in details one of them, namely the number 4, which is the most complex:

$$\underbrace{\{ _ \}^*}_{1} \underbrace{WP|WDT}_{2} \underbrace{\{ _ _ _ _ O, O \}^*}_{3} \underbrace{_ _ _ _ O, !O\#1}_{4} \underbrace{\{ _ _ _ _ M|S\#2 \}^*}_{5} \underbrace{\{ _ \}^*}_{6}$$

Each of the 6 generalized-token of the expression above must match subsequent chunks of the whole question. A question, seen as a sequence of tokens, matches the expression above if it consists of contiguous sub-sequences of tokens with the following properties:

1. any sequence of tokens ($\{ _ \}^*$);
2. a token with the POS tag WP or WDT;

3. any sequence of tokens, whatever their POS tag, lemma, word, and NER are ($_$), without any specific KB annotation and without any specific aggregation function (O);
4. a token with any POS tag, any lemma, any word, and any NER, without any specific KB annotation (O) and with a specific aggregation function (!O). This token is assigned the label 1 (#1);
5. any sequence of tokens with any POS tag, any lemma, any word, and any NER, with a KB annotation that can be a measure (M) or a dataset (S). The type of tag for the aggregation function is not specified, which means it can be anything (in practical case, it will be none). These tokens are assigned the label 2 (#2);
6. any sequence of tokens ($\{ _ \}^*$).

This expression can be matched against several questions, such as: “What is the total aid to the Anti Corruption Commission in the Maldives in 2015?”. In general, questions matching this expression ask for the computation of an aggregation function, which is represented by the token with label 1 computed over a measure, which is represented by the token with label 2. This expression also considers the possibility that the measure is implicitly denoted by the name of the dataset (this can happen, for instance, when the dataset is about a specific measure of a set of observations - *expenditures of town of Cary*), or that the measure is not explicitly mentioned. In this case, we assume that a default measure is defined for each dataset. The default measure is obvious for datasets with one only measure. In case of multiple measures, our system need to manually define the default measure for the dataset. This operation is performed just once, when a new dataset is added to the whole KB.

The questions of this kind can be translated into a SPARQL query with the following structure (template):

```
select <groupByvar> <aggrFunct##1>( ?measure)
where {
  ?obs qb:dataSet <dataset> .
  ?obs <measure##2> ?measure .
  <constraints>
}
<groupBy>
```

where <aggrFunct##1> has to be replaced with the actual aggregation function, which can be derived using the token annotated with label 1, <dataset>

1	RegEx	{_}* _,how _,many {_,_,_,_,O,O}* {_,_,_,_,A D#1}+ {_}* select <groupbyvar> count(distinct ?ans) where { ?obs qb:dataSet <dataset> . ?obs <property##1> ?ans . <constraints> } <groupby>
2	RegEx	{_}* _,how _,much high {_}* select <groupbyvar> sum(?measure) where { ?obs qb:dataSet <dataset> . ?obs <measure> ?measure . <constraints> } <groupby>
3	RegEx	_,when {_}* select distinct ?ans where { ?obs qb:dataSet <dataset> . ?obs < http://linkedspending.aksw.org/ontology/refDate > ?ans . <constraints> }
4	RegEx	{_}* WP WDT {_,_,_,_,O,O}* {_,_,_,_,O,!O#1 {_,_,_,_,M S#2}* {_}* select <groupbyvar> <aggrFunct##1>(?measure) where { ?obs qb:dataSet <dataset> . ?obs <measure##2> ?measure . <constraints> } <groupby>
5	RegEx	{_}* WP WDT {_,_,_,_,O,O}* {_,_,_,_,A D#1}+ {_}* select distinct ?ans where { ?obs qb:dataSet <dataset> . ?obs <measure> ?measure . ?obs <property##1> ?ans . <constraints> } <orderbysummeasure>
6	RegEx	{_}* WP WDT {_,_,_,_,O,O}* {_,_,_,_,M S#2}+ {_}* select <groupbyvar> sum(?measure) where { ?obs qb:dataSet <dataset> . ?obs <measure##2> ?measure . <constraints> } <groupby>
7	RegEx	_,have be _,_,there {_}* ask { ?obs qb:dataSet <dataset> . <constraints> }

Fig. 3. Pattern/Templates used by QA³

must be replaced with the URI of the dataset found in the previous step, `<measure##2>` must be replaced with the actual measure, using tokens labeled with 2 (or with the default measure, if no token has been labeled). Finally, `<constraint>`, `<groupbyvar>` and `<groupby>` must be replaced with the actual variable/clauses (possibly empty), that are derived using the KB tagging, as described in the following.

We remark that this strategy for deriving the SPARQL template is quite general and the definition of new templates is quite simple. Although capturing all the natural language questions is not possible through a finite set of patterns, we empirically found that few expressions are enough to cover most of the questions posed in a typical form (see Section 4).

3.3. Filling the query template using the annotations

The input to this step is the output of the previous two steps, i.e., the SPARQL query template, the dataset to be used, and the tokens with the annotations based on the KB tags. In particular, given the SPARQL template, some portions of the template must be replaced in order to obtain the final query, by using the result of the previous KB tagging. For instance, as stated above, `<dataset>` is replaced with the URI of the dataset found in the first step, `<measure##p>` (where p is a label identifier) is replaced with the actual measure described by tokens with label p (or with the default measure), `<aggregationFunction##q>` has to be replaced with the actual aggregation function, which can be derived using the token annotated with label q . The most interesting part is the construction of the constraints and the group-by clauses. In order to construct the constraints, we observe that

- if a literal is tagged, then it must be connected to the observation through an attribute, which is also reported in the annotation;
- if an entity is tagged, then it must be connected to the observation through a dimension. The dimension could be explicitly tagged in the question, but can be also derived by maintaining an index that maps every entity e to the dimensions which can take e as value;

The substring `<constraints>` of the template can be thus replaced with a SPARQL fragment representing the RDF triple pattern obtained as described above.

Regarding the group-by variable and clause, we observe that a question requiring their use has to contain an attribute or dimension which is not bound to a

value (literal or entity, respectively). Therefore, we can try to find those tokens that are tagged with a dimension/attribute X to which no value is associated. We then replace `<groupbyvar>` with a variable identifier, say `?groupbyvar`, and `?observation` is connected to `?groupbyvar` using the URI of X , and the `<groupby>` placeholder is replaced with `group by ?groupbyvar`. If no such attribute/dimension X can be found, both `<groupbyvar>` and `<groupby>` are replaced with the empty string.

4. Experimental results

We implemented the backend of QA³ [6] using python and Java and a web interface is available at <http://qa3.link/>. A screenshot of the system is available in Fig. 4, where the question *For which account type of whiteacre was spent the most?* is posed to QA³ and the results of each intermediate step are shown. The system finds the correct dataset (`city-of-whiteacre-spending`), executes a group-by query against the SPARQL endpoint and finally provides *capital-outlay-transfer* as the answer (which turns to be correct).

The online system allows to freely type questions, but it also provides the user with a quick selection of the questions of the training and test sets of the task 3 of QALD-6 challenge [2], for which the experimental outcomes are described next.

Fig. 5 reports the number of total questions (column “N”), and the results obtained by QA³ [6] over the two set of questions. Besides the average recall, precision and F1-score computed over all the questions, it is interesting analyzing the accuracy of the internal steps of QA³. In particular, for the training set, over 100 available questions, the correct dataset (column “Correct DS”) is found for 81 questions, the correct set of tags is found for 56 questions, and the correct query is generated for 47 questions. QA³ performed even better on the test set²: over the 50 available questions the correct dataset was found for 42 questions, the correct set of annotations for 31 questions and the correct query for 27 questions.

We believe the results are extremely encouraging because despite the small number of patterns (only the

²Note that this results are not the official ones: they are slightly different since we did some minor improvements on the prototype after the challenge

The screenshot shows the QA³ web interface with the following components:

- Header:** SCAI Scalable Analytics Institute logo and QA³: Statistical Question Answering over RDF Cubes.
- Input:** A text box containing the question "For which account type of whiteacre was spent the most?" and a "Submit" button.
- Step 1:** Dataset: city-of-whiteacre-spending. Annotations table showing a chunk of an account type.
- Step 2:** A regular expression and a SPARQL query to filter and order results by measure.
- Step 3:** A more detailed SPARQL query to select distinct answers from a specific dataset.
- Answer:** The final result: <https://openspending.org/city-of-whiteacre-spending/accounttype/capital-outlay-transfer>

Fig. 4. Screenshot of the QA³ web interface, showing all the steps followed by the system and the final answer for an example question taken from QALD-6

Question Set	N	Correct DS	Correct tags	Correct queries	Recall	Precision	F1-score
Training	100	81	56	47	0.49	0.48	0.48
Test	50	42	31	27	0.57	0.56	0.56

Fig. 5. Results obtained by QA³ (using multiple candidate datasets) on QALD-6 training set and test

System	Processed	Recall	Precision	F-1 (over processed questions)	F-1 Global (overall)
SPARKLIS (used by an expert)	50	0.94	0.96	0.95	0.95
SPARKLIS (used by a beginner)	50	0.76	0.88	0.82	0.82
QA3 (our system, initial tuning)	44	0.62	0.59	0.60	0.53
CubeQA	49	0.41	0.49	0.45	0.44

Fig. 6. Comparison against other QA systems, as reported by the QALD-6 independent competition

7 ones described in Fig. 3), the system was able to provide the correct answer in the majority of the test questions. A direct comparison against the other systems described in Section 5 has been independently performed by the QALD 6 Challenge [2], as reported in Fig. 6. The most performer in this comparison is SPARKLIS³, a system that does not accept natural language questions. Instead, as explained in the next section it uses a faceted search approach, and its performance are therefore dependant on the level of expertise the user has (values for expert and beginner are shown).

To the best of our knowledge, the only two systems that answers free natural language questions over RDF cubes are CubeQA [17], described in the next section, and our QA³. Compared to the state-of-the-art CubeQA, we get an remarkable improvement of $0.62/0.41 = 51\%$ in recall, 20% in precision. This is in part given by the ability of QA³ of self-evaluate the confidence of a computed answer (thanks to the score measure of the tagger), and also by the good expressivity and generality of the template/pattern system.

We also remark that F-1 and F-1 Global, i.e., the F-1 measure computed over all questions (not only those for which the system provides an answer) are respectively 33% and 20% higher than those of CubeQA. These results show that QA³ provides a sensible improvement over the state of the art. System settings and tuning may produce results biased on precision w.r.t. recall or viceversa, e.g. a larger number of candidates to keep or higher thresholds for their scores may lead to higher recall.

5. Related Work

Statistical question answering (SQA), meaning question answering over datacubes, is a very recent research area born thanks to the success of projects such as OpenSpending⁴, that collects datasets on public financial information, and LinkedSpending [1,16], that focuses on representing OpenSpending data through the use of RDF and a W3C standardized vocabulary for datacubes [5].

The intersection between RDF Data Cubes and SQA is sometimes referred to as RDCQA [15]. In [15] the RDCQA benchmark of 100 questions and 50 datasets

used for Task 3 (“Statistical question answering over RDF datacubes”) of the 6th Question Answering on Linked Data (QALD-6) Challenge is also described.

From a general point of view, querying RDF data through a user-friendly interface is an active area of research, with quite different approaches. To the best of our knowledge most of the proposed work can be categorized into one of the following: *Faceted Search*, that consists of an iterative process of adding new applicable constraints to a list of previous results [9,8,11]; *Constrained Natural Languages*, where the query is expressed in natural language through an interface that guides the user to write only a set of possible questions; *Question Answering*, where the input of the system is a free text in natural language [13] (some are based on templates [17]); and *By-Example Structured (BES) Query*, where the user starts from an existing instance that resembles the solution and then a Query-by-Example [24] interface allows to change values to express user’s constraints [7].

Our work falls into the pure *Question Answering* category, as the user can write the query in the way it likes it, while disambiguation and understanding is done later. Although datasets, benchmarks and problem definition have been proposed only recently, there are already two other existing works that focused on Statistical Question Answering over Linked Data. The first system is called *CubeQA* [15], and reached an average 0.4 F-measure at Task3 of QALD-6 challenge, as described in the previous section. The system has a lightweight template module that matches the kind of question (what, how much, when, etc.) against a table that provides the expected result type (countable, uncountable, date, etc.). This improves the results provided by the main step, consisting of parsing the query and creating a tree. Elements of the tree are matched against a pre-computed index that allows to find the most probable dataset and dimensions and generate the corresponding SPARQL query.

In contrast, our approach is extremely general in the first steps, based on entity and property recognition within the query. We do not create a tree, trying instead to tag as much as possible numbers and words (either single or multiple words) in the question. We do this for each possible dataset, using an in-memory index. We then rank solutions according to a score, and the one with higher rank is chosen. Rank increases according to the portion of the question that has been interpreted/understood (tagged); also, ranking decreases on the number and size of recognized entities (multiple word tagging is preferred). We then use an origi-

³In fact, it is a special version of the SPARKLIS system tailored to statistical questions, as described in the related work section.

⁴<https://openspending.org/>

nal template system, with an ad-hoc language, that exploits NLP to express SPARQL templates based on the grammar structure of the query. We also propose partial templates, that (if matched) establish a portion of the SPARQL query (e.g. a new triple pattern, the use of group by over a given attribute, or the use of a specific aggregate function). The matching templates are then used together to generate the final SPARQL query. An interesting fact related to our approach is that each step we run provides insights on how well it has been performed. For instance, if a large portion of the question is not tagged, it means that most of it was not understood and, therefore, chances of obtaining the right answer in the following steps are low. In these cases, we can maximize precision over recall, not providing an answer. The same optimization can be applied for the template-based mechanism, since some templates are mutually exclusive (e.g. either SUM or COUNT can be used as a final aggregate function). Furthermore, the template mechanism is very simple and it easily allows new pattern to be input in order to answer more advanced questions. These techniques result in a sensible increment of the precision and recall of our QA³ over cubeQA, as certified by the QALD-6.

A very different approach is presented in the work by Sébastien Ferré [9,8]. Based on the adaptation [11] of the general purpose Faceted Search system called SPARKLIS [9,8], the system allows users to incrementally add new constraints. In particular, disambiguation and most of the semantically difficult tasks are not done automatically, but left to the user. Although SPARKLIS takes care of also generating a natural language question that helps user to verify that the input constraints are leading to what he is willing to search for, this system cannot be considered a Natural Language approach to Question Answering. One peculiarity of the SPARKLIS system is therefore that achieved precision and recall strongly depend on the user which interacts with the system. According to the QALD-6, the system has been tested by two users with different skill level, and in both cases SPARKLIS achieved a higher F-measure than both cubeQA and our QA³.

6. Discussion

In this section we discuss the lessons we learnt from implementing a system tailored for the specific task of statistical question answering.

First of all, as previous work already noticed [18, 14], there are good reasons why an analytical-aware

representation of the data should be published instead of a flat one. In DBpedia, especially if we focus on the Yago categories, information needed to answer the question "What is the 2014 public transportation budget of Frankfurt?" may be found in triples like the following:

```
:Frankfurt :publicTransportBudget2014Eur
"5.6E7"^^xsd:decimal
```

Unfortunately, this representation may not be useful for statistical reasoning or analytics, e.g. "In which year Frankfurt had the higher public transportation budget" or "What is the average public transportation budget in Germany", because some data needed to filter or to aggregate is "hidden" within the property name.

The adopted solution in literature is an ad-hoc form of reification where a specific OLAP ontology is used [5]. This data model must be taken into account by any question answering system, as the matching between the question and the data follow a different path. In particular, this time answers need to be derived from statistical observations, selecting the appropriate dimensions among many, and values often need to be further processed, e.g., by applying aggregate functions.

In our work instead of adapting an existing QA system, we started from scratch. In our opinion, one of the most important decisions to take while designing a statistical QA system is how to handle the multitude of datasets. Questions in the task 3 of QALD-6 testbed all refers to one dataset per query. In fact, more complex query (such as comparisons among countries or cities) could be answered by two different subqueries, each using one dataset only. Pointing to the right dataset is of paramount importance because the result will be definitely wrong otherwise. We learnt that the first phases of the system, that recognize references to properties and entities within the question, are very important to exclude a vast number of datasets unrelated to the question. Still, given the similarities among some datasets contents (e.g., between `town_of_cary_revenues` and `town_of_cary_expenditures`) the search for the right dataset will contain more than one solution in the first phases of the QA system. Later steps, which actually try to match tagged elements with existing patterns and/or generating the SPARQL query, are useful to further disambiguate. Therefore, in our experience finding the right dataset is something that may involve the whole system, until the last step of generating the results. In fact, getting no results may suggest

a wrong interpretation of the question and a different interpretation leading to another SPARQL query could be then assumed in order to improve the recall.

Another aspect is the trade off between precision and accuracy. Every step in QA³ is associated with an internal score that represents our confidence in that specific decision/interpretation. Going further with low scores leads to higher recall but potentially lower precision, and the viceversa is also true. The system can be therefore tailored for a specific application and/or coupled with a user interface that interacts with the user asking for more (potentially wrong) results.

But perhaps the most important lesson we learnt is that patterns are at the same time very important (if the question contains "average of ...", this must be taken into account) but also that specific patterns may support only few questions. One naive approach may be to generate all possible patterns and generate its natural language representation, then trying the best matching with the current question. The exponential explosion in the search space needs to be addressed, perhaps with original compact representations. As described in the paper, we instead developed small, possibly overlapping patterns that may occur together in the same question. These can be seen as "features" in the question, each associated with a fragment of SPARQL. The combination of the features found in the question leads to the complete SPARQL query, unless the features cannot be combined together (in this case another alternative is taken).

We believe that having a good, comprehensive set of these features, that is, question patterns associated with SPARQL fragments, can lead to high precision *and* recall. Our future work is to extend and improve our manually-crafted set of patterns using machine learning techniques. In particular, the QALD-6 testbed, containing both the natural language questions and the SPARQL queries, can be used as input to statistical learners that can learn, e.g., that "average of ..." can be associated to the SPARQL fragment `AVG (. . .)`, leading to new automatically-learned patterns.

7. Conclusions

In this paper we have presented QA³, a system supporting statistical question answering over RDF data cubes. The system combines the syntactical tags obtained using the Stanford tokenizer and POS tagger with semantic tags derived by QA³ from the available knowledge base. An extensible set of regular expres-

sions, defined over the tokens, is used to detect the query template to be used. The template is then filled in by using the previously found KB tags. We remark that this approach, is effective in this context of RDF data cubes because the meaningful questions naturally tend to follow structured patterns, while most of the effectiveness would be lost in general contexts (e.g., questions posed on DBpedia KB), where the variability of the structure of meaningful question is much higher. The preliminary results are encouraging, and we are currently working on the improvement of all the three steps of the translation process used by QA³. In particular, since the wrong annotations often do not enable to fill in all placeholders in the query template, the performances are likely to be improved by considering additional datasets and tagging results, whenever the choice of one dataset does not yield a valid SPARQL query. This is likely to provide a correct answer to some of the questions for which the correct dataset is assigned the same quality measure as other wrong datasets, and even to some questions for which a wrong dataset is initially detected as the best candidate. Low scores in the first step must be appropriately weighted with the higher scores obtained in the later steps.

The tagging system could be also improved by allowing an approximate matching. This possibility, however, is not obvious since while the approximate matching will improve the recall of the tagging, it could also worsen the precision. A more sophisticated handling of synonyms, taking care of a fuzzy semantic relativity could also improve the answers of questions expressed with different terms w.r.t. the one found in the dataset.

Finally, we are also working on improving the set and internal language of SPARQL templates, in order to enable the use of more advanced SPARQL features (es., the `HAVING` clause, the sub-queries, etc.).

Acknowledgements

This research was supported in part by a 2015 Google Faculty Research Award and NIH 1 U54GM114833-01(BD2K).

References

- [1] LinkedSpending project. <http://linkedspending.aksw.org/>.

- [2] Question Answering over Linked Data (QALD). <http://www.sc.cit-ec.uni-bielefeld.de/qald>.
- [3] Statistical Data and Metadata eXchange. <https://sdmx.org/>.
- [4] The Linking Open Data cloud diagram. <http://lod-cloud.net/>.
- [5] The RDF Data Cube Vocabulary. <https://www.w3.org/TR/vocab-data-cube/>.
- [6] QA³ website. <http://qa3.link/>.
- [7] Maurizio Atzori and Carlo Zaniolo. Swipe: searching wikipedia by example. In *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 309–312, 2012.
- [8] Sébastien Ferré. Expressive and scalable query-based faceted search over SPARQL endpoints. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandečić, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, volume 8797 of *Lecture Notes in Computer Science*, pages 438–453. Springer, 2014.
- [9] Sébastien Ferré. SPARKLIS: a SPARQL endpoint explorer for expressive question answering. In Matthew Horridge, Marco Rospocher, and Jacco van Ossenbruggen, editors, *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014.*, volume 1272 of *CEUR Workshop Proceedings*, pages 45–48. CEUR-WS.org, 2014.
- [10] Sébastien Ferré. SQUALL: the expressiveness of SPARQL 1.1 made available as a controlled natural language. *Data Knowl. Eng.*, 94:163–188, 2014.
- [11] Sébastien Ferré. Sparklis: An expressive query builder for sparql endpoints with guidance in natural language. *Semantic Web*, 7(1):95–104, 2016.
- [12] Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Bürgle, Holger Düwiger, and Ulrich Scheel. Faceted wikipedia search. In *Business Information Systems, 13th International Conference, BIS 2010, Berlin, Germany, May 3-5, 2010. Proceedings*, pages 1–11, 2010.
- [13] L. Hirschman and R. Gaizauskas. Natural language question answering: The view from here. *Nat. Lang. Eng.*, 7(4):275–300, December 2001.
- [14] Konrad Höffner and Jens Lehmann. Towards question answering on statistical linked data. In *Proceedings of the 10th International Conference on Semantic Systems, SEMANTICS 2014, Leipzig, Germany, September 4-5, 2014*, pages 61–64, 2014.
- [15] Konrad Höffner, Jens Lehmann, and Ricardo Usbeck. Cubeqa - question answering on RDF data cubes. In Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981 of *Lecture Notes in Computer Science*, pages 325–340, 2016.
- [16] Konrad Höffner, Michael Martin, and Jens Lehmann. Linked-spending: Openspending becomes linked open data. *Semantic Web*, 7(1):95–104, 2016.
- [17] Konrad Höffner, Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. User interface for a template based question answering system. In Pavel Klinov and Dmitry Mourmoumsev, editors, *Knowledge Engineering and the Semantic Web - 4th International Conference, KESW 2013, St. Petersburg, Russia, October 7-9, 2013. Proceedings*, volume 394 of *Communications in Computer and Information Science*, pages 258–264. Springer, 2013.
- [18] Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Survey on challenges of Question Answering in the Semantic Web. *Semantic Web Journal*, 2016.
- [19] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 423–430, 2003.
- [20] Anca Marginean. Gfmed: Question answering over biomedical linked data with grammatical framework. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, pages 1224–1235, 2014.
- [21] Michael Martin, Konrad Abicht, Claus Stadler, Axel-Cyrille Ngonga Ngomo, Tommaso Soru, and Sören Auer. Cubeviz: Exploration and visualization of statistical linked data. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 219–222, 2015.
- [22] Giuseppe M. Mazzeo and Carlo Zaniolo. Answering controlled natural language questions on RDF knowledge bases. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 608–611, 2016.
- [23] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question answering via phrasal semantic parsing. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8-11, 2015, Proceedings*, pages 414–426, 2015.
- [24] Moshé M. Zloof. Query by example. In *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition, AFIPS '75*, pages 431–438, New York, NY, USA, 1975. ACM.