Towards a Message-Driven Vocabulary for Promoting the Interoperability of Question Answering Systems

Kuldeep Singh*, Andreas Both†, Dennis Diefenbach‡ and Saeedeh Shekarpour§
*Fraunhofer IAIS, Sankt Augustin, Germany, Email: kuldeep.singh@iais.fraunhofer.de
†R & D, Unister GmbH, Leipzig, Germany, Email: andreas.both@unister.de
‡Laboratoire Hubert Curien, Saint-Etienne, France, Email: dennis.diefenbach@univ-saint-etienne.fr
§University of Bonn, Bonn, Germany, Email: shekarpour@uni-bonn.de

***Abstract*—Question answering (QA) is one of the biggest challenges for making sense out of data. Web of Data has attracted the attention of question answering community and recently, a number of schema-aware question answering systems have been introduced. While research achievements are individually significant; yet, integrating different approaches is not possible due to lack of a systematic approach for conceptually describing QA systems.**

In this paper, we present a message-driven vocabulary built upon an abstract level. This vocabulary is concluded from conceptual views of different question answering systems. In this way, we are enabling researchers and industry to implement message-driven QA systems and to reuse and extend different approaches without the interoperability and extension concerns.

***Index Terms*—Semantic Web, Software Reusability, Question Answering, Semantic Search, Ontologies, Annotation Model**

I. Introduction

Web of Data is enormously growing (currently more than 84 billion triples[1]) as well as enterprise data. Still, taking advantage of this rapidly growing data is challenging. Therefore, automatic as well as intelligent approaches are needed to (1) make sense of the data, (2) make the data accessible, and (3) provide easy-to-use interfaces for querying data. Question answering (QA) is multi discipline, and it bridges artificial intelligence, information retrieval, and knowledge base. Recently, the question answering community paid considerable attention for adapting and improving QA systems by taking Web of Data into account. As the result of these attempts, there is an emerging generation of QA systems, which are applied on Web of Data (e.g., [1], [13], [27], [29]).

It is important to note that most of the available QA systems are more focused on implementation details and have limited reusability and extensibility in other QA approaches. Hence, considering the challenges of QA systems there is a need of a generalized approach for architecture or ontology of a QA system and semantic search to bring all state-of-the advancement of QA under a single umbrella. While many of these system achieved significant performance for special use cases, a shortage was observed in all of them. We figured out that the existing QA systems suffer from the following drawbacks: (1) potential of reusing its components is very weak, (2) extension of the components is problematic, and (3) interoperability between the employed components are not systematically defined. Therefore, there is a need for a descriptive approach that define a conceptual view of QA systems. This approach must cover all needs of current QA systems and be abstracted from implementation details. Moreover it must be open such that it can be used in future QA systems. This will allow interoperability, extensibility, and reusability of QA approaches and components of QA systems.

In this paper, we introduce a generalized ontology which covers the need for interoperability of QA systems on a conceptual level. We initiate a step towards a message-driven interoperable approach that will be used to build systems which follow a philosophy of being actually open for extensions. Our approach collects and generalizes the necessitated requirements from the state-of-the-art of QA systems. We model the conceptual view of QA systems using and extending the Web Annotation Data Model. This model empowers us for designing a message-driven approach for QA systems. To the best of our knowledge, in this way we will establish for the first time a conceptual view of QA systems.

The rest of this paper is structured as follows. Section II describes the diverse field of QA systems and its classification. It also covers current approaches for establishing an abstraction of QA systems and their limitations. Section III introduces dimensions of QA based on which many requirements to build open QA systems can be identified. Section IV describes the existing problem and our proposed idea of an implementation independent compatibility level in detail. Section V details the requirement of message-driven QA systems which are derived from the state-of-the-art QA approaches. Section VI illustrates our proposed ontology with a case study to address all the requirements. Conclusion and future work are presented in Section VII.

II. Related Work

In the recent past, different systems for QA have been developed. This section provides a brief overview on various QA system, their scope of applicability, and their different components.

The QA systems can be distinguished based on scope of applicability and approaches. Some of them consider a specific

[1]observed on 14 October 2015 at http://stats.lod2.eu/

domain to answer a query, they are known as *closed domain* QA systems. These QA systems are limited to a specific Knowledge Base (KB), for example medicine [1].

However, when scope is limited to an explicit domain or ontology, there are less chances of ambiguity and high accuracy of answers. It is also difficult and costly to extend closed domain systems to a new domain or reusing it in implementing a new system.

To overcome the limitations of closed domain QA systems, researchers have shifted their focus to *open domain* QA systems.

FreeBase [3], DBpedia [2], and Google's knowledge graph [25] are few examples of open domain Knowledge Bases. Through KBs like Google's knowledge graph are not publically available. Open domain QA systems use publically available semantic information to answer questions.

Other type of QA systems described in [27] extract answers from an unstructured corpus (e.g., news articles), or other various form of documents available over Web. They are known as corpus based QA systems. QuASE [27] is one of such corpus based QA system that mines answers directly from the Web.

In 2011, a yearly benchmark series QALD was introduced. In the latest advancements, QALD now focus on hybrid approaches using information from both structured and unstructured data. Many open domain QA systems now use QALD for the evaluation. PowerAqua [13] is an ontology based QA system which answers the question using the information that can be distributed across heterogeneous semantic resources. FREyA [6] is another QA system that increases system's QA precision by learning user's query formulation preferences. It also focuses to resolve ambiguity while using natural language interfaces. QAKiS [5] is an agnostic QA system that matches fragments of the question with binary relations of the triple store to address the problem of question interpretation as a relation-based match. SINA [23] is a semantic search engine which obtains either keyword-based query or natural language query as input. It uses a Hidden Markov model for disambiguation of mapped resources and then applies forward chaining for generating formal queries. It formulates the graph pattern templates using the knowledge base. TBSL [29] is a template based QA system over linked data that matches a question against a specific SPARQL query. It combines natural Language Processing capabilities (NLP) with linked data to produce good results w.r.t. QALD benchmark.

The field of QA is so vast that the list of different QA systems can go long. Besides domain specific question answering, QA systems can be further classified on type of question (input), sources (structured data or unstructured data), and based on traditional intrinsic challenges posted by search environment (scalability, heterogeneity, openness, etc.) [15]. For a QA system, an input type can be anything ranging from keyword, factoids, temporal and spatial information (e.g., the geo-location of the user), audio, video, image etc. Many systems have been evolved for a particular input type. For example, DeepQA of IBM Watson [19], Swoogle [7], and Sindice [20] focus on keyword-based search whereas systems described in [10] integrates QA and automated speech recognition (ASR). Similarly, there are several examples of QA based on different sources used to generate an answer like Natural Language Interfaces to Data Bases (NLIDB) and QA over free text.

Earlier in this section, we have observed that the field of QA is growing and new advancements are made in each of existing approaches over the short period of time. However, there is a need of an open framework for generating QA system that integrates state-of-the-art of different approaches. Now we discuss some approaches for establishing an abstraction of QA systems and semantic search.

Research presented in [28] describes a search ontology to provide abstraction of users question. User can create complex queries using this ontology without knowing the syntax of the query. This approach provides a way to specify and reuse the search queries but the approach is limited in defining properties represented within the ontology. Using search ontology, user can not define the dataset that should be used and other specific properties. The QALL-ME framework [8] is an attempt to provide a reusable architecture for multilingual, context aware QA framework. It is an open source software package. The QALL-ME framework use an ontology to model structured data of a targeted domain. This framework is restricted to closed domain QA, and finds limitation to get extended for heterogeneous data sources and open domain QA systems.

The openQA [17] framework is an architecture that is dedicated to implement a QA pipeline. Additionally, here the implementation of a QA pipeline is limited to Java and cannot work agnostic to the programming language. For implementing an open framework for generating QA systems, it is important to define a generalized vocabulary for QA. It will be an abstraction level on top of all the existing QA approaches and will provide interoperability and exchangeability between them. This generalized vocabulary can be further used to integrate different components and web services within a QA system. DBpedia Spotlight [18] is an open source web service for annotating text documents with DBpedia resources. AIDA [12] is a similar project which uses the YAGO [26] ontology. While the last two examples address only specific ontologies, AGDISTIS [31] is an approach for name entity disambiguation that can use any ontology. To leverage the capabilities of different available web services and different tools for QA systems, a generalized vocabulary is needed.

III. Dimensions of Question Answering Systems

When we look at the typical pipeline of QA systems, the complete QA process is oriented to three main dimensions as follows: (i) *Query dimension:* covers all processing on input query. (ii) *Answer dimension:* refers to processing on the query answer. (iii) *Dataset dimension:* is related to the both characteristics of and processing over employed datasets. Figure 1 presents these high level dimensions. Generally, all of various known QA processes either associated or interacted with QA systems are corresponding to one of these dimensions. In the following, each dimension is discussed in more detail.

1) *Query Dimension:* The first aspect of this dimension refers to the characteristics of the input query. User-

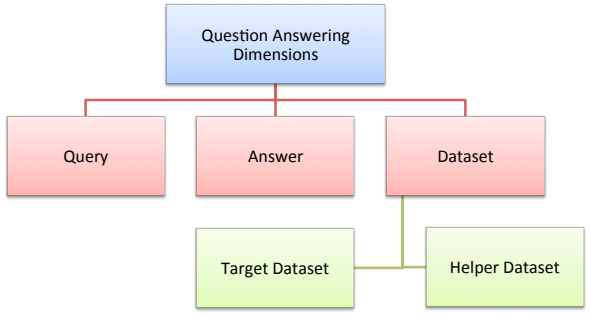

Fig. 1. Main dimensions of question answering systems.

supplied queries can be issued through multifold interface such as voice-based, text-based, and form-based. Apart from the input interface, each query can be expressed in its full or partial form. For instance, full form of a textual or voice query is a complete natural language query whereas partial form is an incomplete natural language query (i.e., either keyword-based query or phrase-based); or full form of a form-based query is a completion of all fields of the form. The second aspect of this dimension refers to processing techniques, which are run on the input query, e.g., query tokenization or Named Entity Recognition.

2) *Answer Dimension:* Similar to the input query, answer dimansion (i.e., refers to the retrieved answer for the given input query) also can have its own characteristics. Answer can have different types (e.g., image, text, audio) also with full or partial form. For instance, a given query can have either a single or a list of items as answer (e.g., the query "islands of Germany" has a list of items as answer). The system might deliver a complete answer (the full list of items) or partial answer (i.e., a subset of items).
3) *Dataset Dimension:* This dimension also has a series of inherent characteristics such as (i) The type of a dataset refers to the format in which s dataset has been published, i.e., structured, semi-structured or unstructured. (ii) Domain of dataset specifies subject of information included. (e.g., movies, sport, life-science and so forth). If the domain of a dataset covers all subjects, the dataset is open domain. In contrast, a closed domain dataset is limited to a few number of subjects. (iii) The size of data obviously shows how big are the employed datasets. Datasets hold two sub-dimensions as follows:

   a) *Helper Dataset Sub-dimension* This dimension includes all datasets required for (i) providing additional information, (ii) training the models. In other words, the helper dataset is used for annotating the input query. Dictionaries like WordNet, gazetteers are examples of this kind of dataset.
   b) *Target Dataset Sub-dimension* Target datasets are leveraged for finding and retrieving answer of input query. For instance, Wikipedia can be a target dataset for search.

IV. Problem and Idea

In this section we will present the problem observed from the state-of-the-art. Thereafter we will outline our idea for solving the problem.

Problem: Considering the related work it is clear that three groups of problems exist:

1) *Lack of a generic conceptional view on QA systems:* While there are many different architectures for QA system (e.g., [27], [10], [17], [13]), most of them are tailored to specific and limited use cases as well as applications. Reusability and interoperability was not (enough) in the focus of such approaches. Creating new QA systems is cumbersome and time consuming as well as limited to domains or programming languages.
2) *No standardized message format for QA systems:* While there are plenty of available tools and services employed by QA systems, yet, interoperability is not ensured due to a missing message format. However, there might be great synergy effect while creating QA systems w.r.t. the combination of different tools. For example, in a given architecture, Named Entity Recognition (NER) and Named Entity Disambiguation (NED) are integrated in a single tool. NER solutions might be evolved and thus, implemented in either a novel way (e.g., in [18]) or employ existing tools (e.g., the Stanford NER [9] used in [12]). Thus, integrating a (new) NER approach without a standardized message format is also cumbersome and time consuming. However, integrating different components is very difficult and causes a lot of work for each new implementation.
3) *Scalability and Coverage problem:* Existing schema-driven approaches are mainly focus on the input query (even limited to textual representations of input query), and aren't flexible for fulfilling emerging demands which are not discovered yet (e.g., [28]).

Hence, considering the implementations of current QA systems (and their components) it can be observed they do not achieve compatibility due to their concentration on the implementation instead of the conceptial view. Instead, implementation details need to be hidden and the focus has to be on the message format communicated between the components of the QA system.

Idea: A conceptual view of QA systems has to be completely implementation-independent. Therefore, we introduce a vocabulary (i.e., schema) that addresses abstract definitions of the data needed for solving QA tasks. We will describe the data model by following the mentioned dimensions (cf., Sec. III).

1) *Input Query:* The abstract definition of the input query along with its properties used for the interpretation and transformation leading towards a formal query of the given dataset.
2) *Answer:* The abstract definition of the answer (i.e., the search result for the given input query) covering all its associated properties.
3) *Dataset:* The abstract definition for all kinds of data being used as background knowledge (i.e., for interpreting

the input query and retrieving the answer).

Please note that we do not describe a specific architecture. Instead our focus is the conceptual level, i.e., the format of the message that needs to be used as input and returned as output by the components of the QA system. Hence, properties need to be *annotated* to the question to make them available for the following components in the QA system pipeline. As each component of the QA system will use the message to retrieve and encode its knowledge about the question from the message, QA applications following this idea are called *message-driven*. Hence, information need to be *annotated* to the message to make it available for following components in the QA system pipeline.

We adapt the definition of annotations from the Web Annotation Data Model[2].

Definition 1 (Annotation): An annotation is an object having the properties *body* and *target*. There should be associated one or more instances of the body property of an annotation object, but there might be zero body instances. There must be one or more instances of the target property of an annotation. For example, considering the question "Where was the European Union founded?" (target) it might be annotated that it contains the named entity "European Union" (body).

In many circumstances, it is required to retrieve the *annotator* (i.e., the creator) of *annotations*. Thus, we demand the provenance of each annotation to be expressable (e.g., while using several NED components and later pick one interpretation). We manifest this in the following requirement:

Req. 1 (Provenance of Annotations): The provenance of each annotation needs to be representable within the data model. The annotator needs to be a resource that identifies the agent responsible for creating the annotation.

Hence, if annotations are available, then each atom of the question can be annotated with additional information.

Definition 2 (Question Atom): The smallest identifiable part of a question (user query) is called question atom and denoted by q_a. Thus, each given user query Q independent of its type consist of a sequential set of atoms $Q = (q_1, q_2, \ldots, q_n)$.

For example, while considering the question to be a text, the characters of the string or the words of the query might be considered as question atoms, while a user query in the form of an audio input (e.g., for spoken user queries) might be represented as byte stream. Considering textual questions, main component might be parser or Part of Speech taggers. They are used to identify relations between the terms in a question. These relations can have a tree structure like in the case of dependency trees but also more complex ones like direct acyclic graphs (DAG) that are used for example in Xser [33]. Some QA systems such as gAnswer [34] use coreference resolution tools, i.e., finding phrases that refer to some entity in the question. Moreover, a typical goal of QA systems is to group phrases in triples that should reflect the RDF structure of the given dataset. These examples imply the following requirement:

[2]W3C First Public Working Draft 11 December 2014, http://www.w3.org/TR/annotation-model/

Req. 2 (Relations between Annotations): (a) It has to be possible to describe relations between annotations. (b) Using these relations, it has to be possible to describe a directed or undirected graph (of annotations).

Annotations of components do not always have boolean characteristics. It is also possible to assign a confidence, (un)certainty, probablity, or (in general) score for the annotations. Once again an example is the NED process, where for entity candidates also a certainty is computed (like in [18]). This implies the following requirement:

Req. 3 (Score of Annotations): It should be possible to assign a *score* to each annotation.

Note: The type of the score is an implementation detail, e.g., in [18] the confidence (score) is within the range $[0, 1]$ while in [21] the score might be any decimal number.

In the next section the three main dimensions of QA system are described and necessary requirements are derived.

V. Requirements of Message-driven Approach

In this section while aiming for a conceptional level the requirements of message-driven approach for describing QA systems are derived from the state-of-the-art (cf., Section II). We present them following the dimensions of QA systems as described in Section III. Hence, on the one side a data model for messages in QA systems should be able to describe at least actual QA systems. On the other side the data model has to be flexible and extensible since it is not known how future QA systems will look like nor which kind of annotations their components will use. In general, there are two main attributes which we have to take into account:

- The proposed approach should be comprehensive in order to catch all known annotations used so far in QA systems.
- It should have enough flexibility for future extensions in order to be compatible with the upcoming annotations.

A. Input Query

The input query of a QA system can be of various types. For example it might be a query in natural language text (e.g., [30]), a keyword-based query (e.g., [24]), an audio stream (e.g., [16]), or a resource-driven input (e.g., [4]). In all these cases the parts of an input query need to be identifiable as a referable instance such that they can be annotated during the input query analysis. Hence, we define the following requirement:

Req. 4 (Part of the Input Query): The definitions of *part of the input query* satisfies the following conditions: (i) *Part consists of a non-empty set of parts and atoms:* each part might be an aggregation of atoms and other parts. However, the transitive closure of the aggregation of each part needs to contain at least one question atom. (ii) For an input query an arbitrary number of *parts* can be defined.

Please note that as we mentioned before, all of the requirements or definitions are independent of any implementation details. Thus, for instance, input query, atom or part have to been interpreted conceptually.

Examples from an implementation view are as follows: for text queries the NIF [11] vocabulary might be used to

identify each part by its start and end position within the list of characters. Similarly, the Open Annotation Data Model [22] selects parts of a binary stream by indicating the start and end position within the list of bytes. We leave the actual types of atoms and properties of parts open, as it is depending on the implementation of the actual QA system.

In a QA system the components of the analytics pipeline will annotate the parts of the query. Examples for such components are Part-of-Speech (POS) taggers (e.g., in [14]), Named Entity Recognition (NER) tools (e.g., in [5]) or Named Entity Disambiguation (NED) tools (like [31]). One possible scenario for a textual question is that first several parts are computed (e.g., with a POS-tagger). Thereafter a NED component might annotate the parts with the additional properties, expressing the current state of the question analytics, using the properties that are accepted in the NED community. As it is not known what kind of properties are annotated by which component, we will not define them here. Hence, we have to keep the annotations open for on-demand definition:

Req. 5 (Annotations of Parts): It has to be possible to define an arbitrary number of annotations for each part.

For example, for the input textual query "capital of Germany", the part "Germany" might be annotated by a NER tool as place (e.g., while using `dbo:place`[3]).

B. Answer

Each QA system is aiming at the computation of a result. However, considering the QA task there are some demands of the type of the answer. For example, the QA task might demand a boolean answer (e.g., for "Did Napoleons first wife die in France?"), a set of resources (e.g., for "Which capitals have more than 1 mio. inhabitants?"), a list of resources, just one resource etc. Therefore, we define the following requirement:

Req. 6 (Answer): The question needs to be annotated with an object typed as answer. A resource of the type answer might be annotated with a property describing the type of the QA task (e.g., boolean, list, set, ...).

Of course, it is possible that the answer type is preconfigured by the user as well as that it needs to be derived automatically by the QA system from the given question.

Additionally only several types might be acceptable for the items been contained in the answer. For example, given the question "Who was the 7th secretary-general of the United Nations?" only specific resource types are acceptable as answer items (w.r.t. the given data). Here it might be `dbo:person`. From this observation we derive the following requirement.

Req. 7 (Types of Answer Items): An arbitrary number of types can be annotated, to express the types acceptable for the items within the answer.

As an answer is also an annotation of the question, the answer, its answer item types and any additional pieces of information might also be annotated with a score (cf., Req. 3), the annotator (cf., Req. 1) etc.

[3]`@prefix dbo: <http://dbpedia.org/ontology/>`

C. Dataset

The proposed data model needs to take into account information about the employed datasets. The dataset dimension and its sub-dimensions were introduced in the section III. To include these dimensions to the data model, the following requirements are met:

Req. 8 (Dataset): A *dataset* provides an *endpoint* where the data can be accessed and statements about the dataset *format* can be gathered.

Req. 9 (Helper Dataset): A question should be annotated by an arbitrary number of helper datasets (which are subclass of dataset class).

Req. 10 (Target Dataset): At least there is one target dataset (which is subclass of dataset class). Both question and answer should be annotated by at least one target dataset (number of target datasets is arbitrary).

These requirements enables QA system components to easily (1) spot data, (2) access data, (3) query data. Please note that target datasets might overlap with the helper data sets and vice versa.

VI. Case Study

In the previous section, we have collected the requirements for a data model describing the message of interoperable QA systems. As case study we know focus on an ontology that fulfills these requirements (although other formal representation will also comply with the requirements). Here, the Web Annotation Data Model (WADM[4]) is used as basis that is currently a W3C working draft. The WADM is an extensible, interoperable framework for expressing annotations and is well accepted. In the following it is shown that how the identified requirements are met.

The WADM introduces the class `Annotation` of the vocabulary `oa`[5]. Thus, any annotation can be defined as an instance of this class. The class `Annotation` has two major characteristics as the body and the target. The body is "about" the target and it can be changed or modified according to the intention of the annotation. The basic annotation model is represented in Turtle format[6] as follows:

```
<anno> a            oa:Annotation ;
       oa:hasTarget <target>      ;
       oa:hasBody   <body>        .
```

The above pseudocode describes an annotation instance which is identified by `anno`. The `anno` has the properties `target` and `body` (i.e., each one is a resource). In the following, we extend the WADM in order to meet all requirements. For this purpose, a new namespace is introduced:

```
@prefix qa: <urn:qa> .
```

In order to illustrate the implications, we use a running example with the question "Where was the European Union founded?". First an instance with the type `qa:Question` is

[4]W3C First Public Working Draft 11 December 2014, http://www.w3.org/TR/annotation-model/

[5]`@prefix oa: <http://www.w3.org/ns/oa#> .`

[6]http://www.w3.org/TR/owl2-manchester-syntax/RDF 1.1 Turtle, W3C Rec. 2014-02-25, http://www.w3.org/TR/turtle/

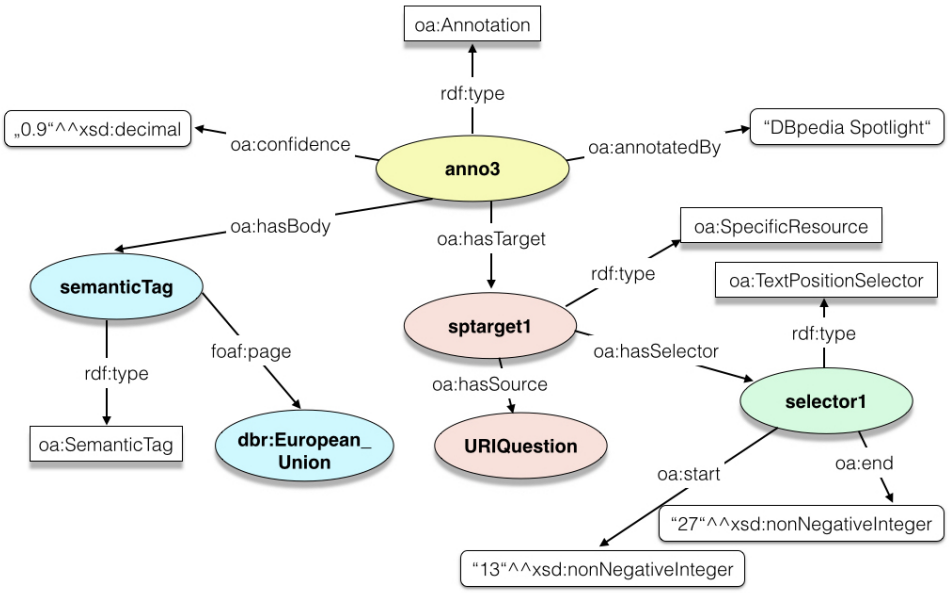

Fig. 2. This picture represents an annotation of the question ”Where was the European Union founded?”. The part ”European Union” is selected using a Specific Resource and a Selector. Moreover a semantic tag is associated to it.

instantiated with the identifier `URIQuestion`. We extended the data model as the input query needs to be defined as well as the answer and the dataset. These concepts are represented by the classes `qa:Question`, `qa:Answer` and `qa:Dataset` which are used to identify questions, answers and datasets. For the example also a URI for the answer `URIAnswer` and for the dataset `URIDataset` is introduced. Then one can establish the corresponding instances:

```
<URIQuestion>   a   qa:Question .
<URIAnswer>     a   qa:Answer   .
<URIDataset>    a   qa:Dataset  .
```

These annotations instantiates question, answer and dataset object. To establish annotation of a question instance we introduce a new type of annotation namely `qa:AnnotationOfQuestion`. It is defined as follows:

```
qa:AnnotationOfQuestion rdf:type owl:Class    ;
                rdfs:subClassOf oa:Annotation  ;
                owl:equivalentClass [
                  rdf:type owl:Restriction     ;
                  owl:onProperty oa:hasTarget  ;
                  owl:someValuesFrom qa:Question
                ].
```

This means that annotations of this type need to have a target of type question. Analogously two new annotation types are introduced `qa:AnnotationOfAnswer` and `qa:AnnotationOfDataset`. In our example the question is annotated with an answer (`anno1`) and a dataset (`anno2`).

```
<anno1> a            oa:AnnotationOfQuestion ;
        oa:hasTarget <URIQuestion> ;
        oa:hasBody   <URIAnswer>   .
<anno2> a            oa:AnnotationOfQuestion ;
        oa:hasTarget <URIQuestion> ;
        oa:hasBody   <URIDataset>  .
```

Now, we will consider requirement 4. To select parts of a query the WADM introduces two concepts: *Specific Resources* and *Selectors*. In the WADM, there is a class called Specific Resource (`oa:SpecificResource`) for describing a specific region of another resource called source. We use this class for typing the concept of part of query in our data model. Assume “European Union” is a part of the input query. For this part, we instantiate a resource with the identifier `sptarget1` and the type `oa:SpecificResource`. The WADM introduces the property `oa:hasSource` which connects a specific resource to its source. In our example, the source of `sptarget1` is `URIQuestion` stating that “European Union” is a part of the input query. Another relevant class which can be captured from the WADM is the class `oa:Selector`. It describes how to derive the specific resource from the source. In our example we instanciate the resource `selector1` which is a particular type of selector, namely a `oa:TextPositionSelector`. It descibes that the part “European Union” can be selected in the input query between the character 13 and 27. This is indicated using the properties `oa:start` and `oa:end`. This can be expressed via:

```
<sptarget1> a              oa:SpecificResource;
            oa:hasSource   <URIQuestion>;
            oa:hasSelector <selector1> .
<selector1> a              oa:TextPositionSelector;
            oa:start       13 ;
            oa:end         27 .
```

WADM introduces other types of selectors like *Data Position Selectors* for byte streams and *Area Selectors* for images. Hence, the requirement 4 is fulfilled. It is obvious that we can instantiate an arbitrary number of annotations for each part of a question. Thus, the requirement 5 is also met.

The WADM defines the property `oa:annotatedBy` to identify the agent responsible for creating the Annotation, s.t., requirement 1 is fulfilled. To comply with requirement 3 a new property `qa:score` with domain `oa:Annotation`

and range xsd:decimal is introduced. For example, if "European Union" is annotated by DBpedia Spotlight[7] with a confidence (score) of 0.9, this can be expressed as:

```
<anno3>       a                 oa:Annotation      ;
              oa:hasTarget      <sptarget1>        ;
              oa:hasBody        <semanticTag>      .
<semanticTag> a                 oa:SemanticTag     ;
              foaf:page         dbr:European_Union .
<anno3>       oa:annotatedBy    DBpedia spotlight  ;
              oa:score          "0.9"^^xsd:decimal .
```

To fulfill requirement 6, in our data model a new class qa:AnswerFormat and a new type of annotation qa:AnnotationOfAnswerFormat are introduced:

```
qa:AnnotationOfAnswerFormat a   owl:Class        ;
            rdfs:subClassOf oa:AnnotationOfAnswer;
            owl:equivalentClass [
              rdf:type           owl:Restriction ;
              owl:onProperty     oa:hasBody      ;
              owl:someValuesFrom qa:AnswerFormat
            ].
```

If the expected answer format is a string, then this can be expressed with the following annotation:

```
<anno4>  a            qa:AnnotationOfAnswerFormat ;
         oa:hasTarget <URIAnswer> ;
         oa:hasBody   <body4> .
<body4>  a            qa:AnswerFormat ;
         rdfs:label   "String" .
```

Although later a resource will be used (instead of the rdfs:label), this shows that the requirement 6 is met. Requirement 7 is analogously satisfied. Now the requirements for datasets are considered. To fulfill requirement 8 a new class qa:Endpoint is introduced having the property qa:hasFormat and a new annotation qa:AnnotationOfEndpointOfDataset:

```
qa:AnnotationOfEndpointOfDataset a owl:Class ;
    rdfs:subClassOf oa:AnnotationOfDataset   ;
    owl:equivalentClass [
      rdf:type owl:Restriction  ;
      owl:onProperty oa:hasBody ;
      owl:someValuesFrom qa:Endpoint
    ].
```

If the question in the example should be answered using a SPARQL endpoint available under the URI body5 (e.g., http://dbpedia.org/sparql), this might be expressed by:

```
<anno5> a            oa:AnnotationOfEndpointOfDataset;
        oa:hasTarget <URIDataset> ;
        oa:hasBody   <body5>      .
<body5> a            qa:Endpoint  ;
        qa:hasFormat dbr:SPARQL   .
```

To fulfill requirements 9 and 10 two new classes are introduced qa:HelperDataset and qa:TargetDataset. Both are subclasses of qa:Dataset. If DBpedia is considered to be a target dataset, this can be expressed as follows while URIDataset is http://dbpedia.org:

```
<URIDataset> a          qa:TargetDataset       ;
             rdfs:label "DBpedia version 2015" .
```

Relations between two annotations <annoX> and <annoY> with a label can be expressed using a new annotation in the following way:

```
<annoZ> a            oa:Annotation ;
        oa:hasTarget <annoX>       ;
        oa:hasBody   <annoY>       ;
        rdfs:label   "my anotation label" .
```

This corresponds to a directed edge between the two annotations. Representing undirected edges is possible in a similar way. This shows that requirement 2 is also fulfilled.

To sum up, in this case study we expressed the demanded datamodel with a generalized ontology which is reusing the concepts of the Web Annotation Data Model. We have shown that it is able to address all the requirements identified in Section V. Moreover, we have illustrated the usage with an example. While using annotation, the data model is extensible and open for improvements. The complete case study is available as online appendix at https://goo.gl/vECgK5.

VII. Conclusion and Future Work

In this paper we have motivated the high demand for an ontology which covers the need for interoperability of QA systems on a conceptual level. We distinguish our approach from other attempts to establish a QA system architecture. Instead we focus on the message level, s.t., everything needed to establish a QA system is included within the data model. Given the requirements and the corresponding case study to the best of our knowledge, we have established for the first time a message-driven interoperable approach that follows a philosophy aiming for QA systems actually open for extension.

Consequently, we collected the requirements for the data model from the recent works to cover also the needs of existing QA systems. However, previous work consider only specific scopes of applicability of QA systems (particularly many focus on text-driven QA). We have chosen an approach that is agnostic to the implementation and the actual representation of the input question and the answer as well as the data sets. Hence, it is a descriptive and open approach. This is an important milestone on the way to actual open QA systems.

We have shown in our case study that our requirements can be covered by the Web Annotation Data Model. Hence, a logical, extensible, and machine-readable representation is now available fulfilling the collected requirements. Eventually the proposed approach can be used for all the existing QA systems while transforming them into a message-driven (and therefore interoperable) implementations.

We see this work as the first step in a larger research agenda. Based on the presented data model (or its implementation as an ontology) the research community is enabled to establish a new generation of QA systems and components of QA systems that are interoperable. Hence, actually open QA systems are in sight. Based on our contribution the research community and the industry can work with a best-of-breed paradigm while establishing future QA systems and integrate novel components into their QA systems. Additionally, our approach enables developers to integrate several components with the same task (e.g., NED) and thereafter compare the results (e.g., with ensemble methods) to achieve the best results w.r.t. the considered domain or data. Consequently we will also aim at an implementation of an open QA system that can be used for comparable benchmarks (like it was done for entity

[7] @prefix dbr: <http://dbpedia.org/resource/>

annotation in [32]) strengthening the competition of different approaches as well as measuring the influence of different data sets.

Acknowledgments Parts of this work received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 642795 (WDAqua project). We thank Didier Cherix for his kind support while developing the approach.

REFERENCES

[1] Asma Ben Abacha and Pierre Zweigenbaum. Medical question answering: translating medical questions into sparql queries. In *ACM International Health Informatics Symposium, IHI '12, Miami, FL, USA, January 28-30, 2012*, pages 41–50, 2012.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735, 2007.

[3] Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. Freebase: A shared database of structured general human knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1962–1963, 2007.

[4] Andreas Both, Viet Nguyen, Mandy Keck, Dietrich Kammer, Rainer Groh, and Dana Henkens. Get inspired: A visual divide and conquer approach for motive-based search scenarios. In *13th International Conference WWW/INTERNET (ICWI 2014)*. International Association for Development of the Information Society (IADIS), 2014.

[5] Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. Qakis: an open domain QA system based on relational patterns. In *Proceedings of the ISWC 2012 Posters & Demonstrations Track, Boston, USA*, 2012.

[6] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Freya: An interactive way of querying linked data using natural language. In *The Semantic Web: ESWC 2011 Workshops, Heraklion, Greece, Revised Selected Papers*, pages 125–138, 2011.

[7] Li Ding, Timothy W. Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA*, pages 652–659, 2004.

[8] Óscar Ferrández, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrández, Matteo Negri, Rubén Izquierdo, David Tomás, Constantin Orasan, Guenter Neumann, Bernardo Magnini, and José Luis Vicedo González. The QALL-ME framework: A specifiable-domain multilingual question answering architecture. *J. Web Sem.*, 9(2):137–145, 2011.

[9] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, 2005.

[10] Sanda M. Harabagiu, Dan I. Moldovan, and Joe Picone. Open-domain voice-activated question answering. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan*, 2002.

[11] Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating NLP using linked data. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, pages 98–113, 2013.

[12] Yusra Ibrahim, Mohamed Amir Yosef, and Gerhard Weikum. Aida-social: Entity linking on the social stream. In *Proceedings of the 7th International Workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR '14, Shanghai, China*, pages 17–19, 2014.

[13] Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265, 2011.

[14] Vanessa Lopez, Enrico Motta, Marta Sabou, and Miriam Fernandez. Poweraqua: A multi-ontology based question answering system–v1. *OpenKnowledge Deliverable D8. 4 Pp1*, 14, 2007.

[15] Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. Is question answering fit for the semantic web? a survey. *Semantic Web*, 2(2):125–155, 2011.

[16] Jordi Luque, Daniel Ferrés, Javier Hernando, José B Mariño, and Horacio Rodríguez. Geovaqa: a voice activated geographical question answering system.

[17] Edgard Marx, Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Konrad Höffner, Jens Lehmann, and Sören Auer. Towards an open question answering architecture. In *Proceedings of the 10th Int. Conf. on Semantic Systems, SEMANTiCS 2014, Leipzig, Germany*, pages 57–60, 2014.

[18] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings the 7th Int. Conf. on Semantic Systems, I-SEMANTICS 2011, Graz, Austria*, pages 1–8, 2011.

[19] Alessandro Moschitti, Jennifer Chu-Carroll, Siddharth Patwardhan, James Fan, and Giuseppe Riccardi. Using syntactic and semantic structural kernels for classifying definition questions in jeopardy! In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 712–724, 2011.

[20] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52, 2008.

[21] Stefan Ruseti, Alexandru Mirea, Traian Rebedea, and Stefan Trausan-Matu. Qanswer-enhanced entity matching for question answering over linked data. CLEF, 2015.

[22] Robert Sanderson, Paolo Ciccarese, Herbert Van de Sompel, Shannon Bradshaw, Dan Brickley, Leyla Jael García Castro, Timothy Clark, Timothy Cole, Phil Desenne, Anna Gerber, et al. Open annotation data model. *W3C Community Draft*, 2013.

[23] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. SINA: semantic interpretation of user queries for question answering on interlinked data. *J. Web Sem.*, 30:39–51, 2015.

[24] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sren Auer. Sina: Semantic interpretation of user queries for question answering on interlinked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 30:39 – 51, 2015. Semantic Search.

[25] Amit Singhal. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*, 2012.

[26] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706, 2007.

[27] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy*, pages 1045–1055, 2015.

[28] Alexandr Uciteli, Christoph Goller, Patryk Burek, Sebastian Siemoleit, Breno Faria, Halyna Galanzina, Timo Weiland, Doreen Drechsler-Hake, Wolfram Bartussek, and Heinrich Herre. Search ontology, a new approach towards semantic search. In *Workshop on Future Search Engines at 44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Stuttgart, Deutschland*, pages 667–672, 2014.

[29] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France*, pages 639–648, 2012.

[30] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st World Wide Web Conference 2012*, pages 639–648. ACM, 2012.

[31] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. AGDISTIS - graph-based disambiguation of named entities using linked data. In *The Semantic Web - ISWC 2014 - 13th Int. Semantic Web Conference, Riva del Garda, Italy. Proceedings, Part I*, pages 457–471, 2014.

[32] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. GERBIL – general entity annotation benchmark framework. In *24th WWW conference*, 2015.

[33] Kun Xu, Yansong Feng, and Dongyan Zhao. Xser@ qald-4: Answering natural language questions via phrasal semantic parsing, 2014.

[34] Lei Zou, Ruizhe Huang, Haixun Wang, Jeffer Xu Yu, Wenqiang He, and Dongyan Zhao. Natural language question answering over rdf: a graph data driven approach. In *Proc. of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM, 2014.