The Qanary Ecosystem: getting new insights by composing Question Answering pipelines

Dennis Diefenbach[1], Kuldeep Singh[2,3], Andreas Both[4], Didier Cherix[5], Christoph Lange[2,3], and Sören Auer[2,3]

[1] Laboratoire Hubert Curien, Saint Etienne, France,
dennis.diefenbach@univ-st-etienne.fr
[2] Fraunhofer IAIS, Sankt Augustin, Germany
kuldeep.singh@iais.fraunhofer.de
[3] University of Bonn, Bonn, Germany
{langec|auer}@cs.uni-bonn.de
[4] DATEV eG, Germany
contact@andreasboth.de
[5] FLAVIA IT-Management GmbH, Germany
didier.cherix@gmail.com

Abstract. The field of Question Answering (QA) is very multi-disciplinary as it requires expertise from a large number of areas such as natural language processing (NLP), artificial intelligence, machine learning, information retrieval, speech recognition and semantic technologies. In the past years a large number of QA systems were proposed using approaches from different fields and focusing on particular tasks in the QA process. Unfortunately, most of these systems cannot be easily reused, extended, and results cannot be easily reproduced since the systems are mostly implemented in a monolithic fashion, lack standardized interfaces and are often not open source or available as Web services. To address these issues we developed the knowledge-based Qanary methodology for choreographing QA pipelines distributed over the Web. Qanary employs the `qa` vocabulary as an exchange format for typical QA components. As a result, QA systems can be built using the Qanary methodology in a simpler, more flexible and standardized way while becoming knowledge-driven instead of being process-oriented. This paper presents the components and services that are integrated using the `qa` vocabulary and the Qanary methodology within the Qanary ecosystem. Moreover, we show how the Qanary ecosystem can be used to analyse QA processes to detect weaknesses and research gaps. We illustrate this by focusing on the Entity Linking (EL) task w.r.t. textual natural language input, which is a fundamental step in most QA processes. Additionally, we contribute the first EL benchmark for QA, as open source. Our main goal is to show how the research community can use Qanary to gain new insights into QA processes.

Keywords: Semantic Web, Software Reusability, Question Answering, Service Composition, Semantic Search, Ontologies, Annotation Model

1 Introduction

The amount of data, information, and knowledge available on the Web and within enterprise environments is increasing constantly. Especially in enterprise environments a strong trend to better connected data can be observed, leading to interlinked and accessible data unlocking the company's information for intense data analytics and information retrieval. Novel interfaces are required for enabling users to retrieve information in such scenarios and interact with it. Natural language interfaces are being considered to bridge the gap between large amounts of (semi-structured) data and users' needs. Recent industrial applications show the capabilities and advantages of natural language interfaces in the field of Question Answering (QA). These include *Apple Siri*[6], *Microsoft Cortana*[7], and "Ok Google"[8]. However, these proprietary platforms do not facilitate experimentation with cutting-edge research approaches, they offer only limited interfaces for integrating third-party components and they are generally not open, reusable and extensible by developers and the research community.

Several QA systems have been developed recently in the research community, for example, [15,9,5,6]. These systems perform well in specific domains, but their reusability for further research is limited because of their focus on specific technologies, applications or datasets. As a result, creating new QA systems is currently still cumbersome and inefficient. Particularly, the research community is not empowered to focus on improving particular components of the QA process, as developing a new QA system and integrating a component is extremely resource-consuming. Some first steps for developing flexible, modular QA systems have started to address this challenge, e.g., [11,7]. However, these approaches lack several key properties required for constructing QA systems in a community effort as they are, for example, bound to a particular technology environment and have rather static interfaces, which do not support the evolution of the inter-component data exchange models. For this reason we presented the `qa` vocabulary [16] as a flexible and extensible data model for QA systems. Based on the vocabulary, we developed the Qanary [3] methodology for integrating components into QA systems; it is independent from programming languages, agnostic to domains and datasets, as well as enabled for components on any granularity level within the QA process.

This work presents the Qanary *ecosystem*: the components and services currently implemented around the `qa` vocabulary by using the Qanary methodology. We present a general workflow that can be used to construct and particularly analyze as well as optimize future QA systems in a community effort using the Qanary ecosystem. It can be broken down into two phases: (1.) the identification and integration of existing state-of-the-art approaches to solve a particular task in the QA pipeline, and (2.) the derivation of benchmarks for sub-tasks of a QA process from well-known QA benchmarks such as the Question Answering over

[6] `http://www.apple.com/ios/siri/`
[7] `http://windows.microsoft.com/en-us/windows-10/getstarted-what-is-cortana`
[8] `https://support.google.com/websearch/answer/2940021?hl=en`

Linked Data (QALD) challenge[9]. Hence, the described approach is dedicated to support the engineering process to build components for a QA system and the system by itself, by using the knowledge-driven approach for flexible component integration and quality evaluations. In this paper, we show this workflow applied to the task of EL, which is key in the QA process. Therefore, we consider components dedicated to the tasks of named entity identification/recognition (NER) and named entity disambiguation (NED), which are integrated into the Qanary ecosystem. The included components are the NER tool of *DBpedia Spotlight* [12], the *Stanford NER tool* [8] and the *Federated knOwledge eXtraction Framework* (FOX) [18] as well as the NED components *Agnostic Disambiguation of Named Entities Using Linked Open Data* (AGDISTIS) [20] and the named entity disambiguator of DBpedia Spotlight. In addition two combined approaches for NER and NED are also provided as components: *IBM Alchemy*[10] and *Lucene Linker* – a component that we implemented following the idea of the QA system *SINA* [15]. Moreover, we devised a benchmark for entity linking (EL) based on the well-known Question Answering over Linked Data (QALD) challenge. Our contribution here has three aspects. First, we provide researchers with a tool for comparing NED and NER w.r.t. QA, thus enabling them to compare their components with the state-of-the-art just by implementing a Qanary wrapper around their novel functionality. Second, we provide the results of comparing existing tools, i.e., an expressive benchmark for the quality of entity linking components w.r.t. natural language questions, thus enabling the QA community to gain new insights into QA processes. Third, we compute a list of questions that are completely annotated w.r.t. the entity linking process. Hence, researchers investigating a processing step of a QA system that comes after entity linking can reuse these annotations to create an environment for conveniently testing and continuously improving their components.

As a result, the QA community is empowered to easily reuse entity linking functionality for QA systems (or for the development of other tools depending on named entities) and reuse a profound benchmark for QA systems both for the evaluation of new entity linking components and as input for components active in the subsequent processing steps of a QA system (e.g., relation detection or query computation). However, the entity linking functionality and experiments presented in this paper are just a proof that Qanary's knowledge-driven and component-driven approach as well as the previously described general workflow provides key advantages particularly in contrast to existing systems and other benchmarks.

The next section describes related work. Sec. 3 gives an overview of our recent work which laid the groundwork for the Qanary ecosystem. Sec. 4 gives an overview of the components and services that are available in the Qanary ecosystem. Sec. 5 describes how the Qanary ecosystem can be used to gain new insights into QA processes w.r.t. the EL task. Sec. 6 concludes and points to future research areas.

[9] `http://greententacle.techfak.uni-bielefeld.de/cunger/qald`
[10] `http://www.alchemyapi.com/`

2 Related Work

In the context of QA, a large number of systems and frameworks have been developed in the last years. This can be observed for example from the number of QA systems ($>$ 20 in the last 5 years) that were evaluated against the QALD benchmark. Many QA systems use similar techniques. For example, there are services for named entity identification (NER) and disambiguation (NED) such as DBpedia Spotlight [12] and Stanford NER [8], which are reused across several QA systems. These reasons led to the idea of developing component-based frameworks that make parts of QA systems reusable. We are aware of three frameworks that attempt to provide a reusable architecture for QA systems. The first is QALL-ME [7] which provides a reusable architecture skeleton for building multilingual QA systems. The second is openQA [11], which provides a mechanism to combine different QA systems and evaluate their performance using the QALD-3 benchmark. The third is the Open KnowledgeBase and Question-Answering (OKBQA) challenge[11]. It is a community effort to develop a QA system that defines rigid JSON interfaces between the components. Differently from these works we do not propose a rigid skeleton for the QA pipeline, instead we allow multiple levels of granularity and enable the community to develop new types of pipelines.

Recognizing named entities in a text and linking them to a knowledge base is an **essential task** in QA. DBpedia Spotlight [12], Stanford NER [8], FOX [18], and Alchemy API are a few of the tools dedicated to such tasks. Furthermore, tools such as DBpedia Spotlight, AGDISTIS [20], Alchemy API etc. not only identify information units in text queries but also point every named entity to a knowledge resource for disambiguation.

We are not aware of any work that has tried to compare in a systematic way existing approaches that tackle sub-processes of QA pipelines, for example EL. Atdag and Labatut [1] compare a few NER tools applied to bibliographic text, whereas researchers in [14] present NERD, a framework for evaluating NER tools in the context of Web data where a wrapper of NER/NED services was implemented but the independent registration of new services is not possible. Platforms such as GERBIL [21] and GERBIL for QA[12] offer benchmarks for EL tools and full QA systems and they generate persistent URIs for experiment results. This enables third-party evaluations and citable URIs. Their main goal is not to gain new insights into the underlying processes but only to generate one final metric that is publishable. For example, they do not generate a summary indicating in which cases the corresponding tool succeeded or failed. In contrast, the Qanary reference implementation is a full-featured framework for component-oriented QA process creation, which is *additionally* enabled to support benchmarking of the included distributed components. We give a comparison of the described tools in Table 1.

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

[12] `https://github.com/TortugaAttack/gerbil-qa`

Property vs. Tool	NERD	Gerbil	Gerbil-qa	Qanary
support for analyzing NER/NED task	■	■	□	■
support for analyzing QA process quality	□	□	▨	■
third party evaluations	□	■	■	■
fine-grained information	■	□	□	■
traceability of intermediate results	▨	□	□	■
computation of citable URIs	□	■	■	□

Table 1. Overview of tools related to benchmarks in the field of question answering in comparison to the benchmark functionality of Qanary.

3 The qa Vocabulary and the Qanary Methodology

To advance the QA process, researchers are combining different technologies to optimize their approaches. However, reusability and extensibility of QA components and systems remains a major hurdle. There are many components and services, which are provided as standalone implementations but can be useful in QA processes (e.g., the previously mentioned DBpedia Spotlight, AGDISTIS etc.), but there has so far not been a methodology to integrate them within QA pipelines. Instead substantial programming efforts had to be invested as each component provides its own API or integration method.

To address this challenge, and to promote reusability and extensibility of QA components, we introduced the `qa` vocabulary [16]. This vocabulary can represent information that is generated during the execution of a QA pipeline when processing a question given as speech or text input. Consider, for example, the question "When was Barack Obama born?". Typical information generated by components of a QA pipeline are the positions of named entities (NE) (such as "Barack Obama"), the ontological relations used to express the relational phrase in the question (that "born" refers to `dbo:birthPlace`[13]), the expected answer type (here: a date), the generated SPARQL query, the language of the question and possible ontologies that can be used to answer it.

The rationale of `qa` is that all these pieces of information can be expressed as annotations to the question. Hence, these exposed pieces of information can be provided as an (RDF) knowledge base containing the full descriptive knowledge about the currently given question.

`qa` is built on top of the *Web Annotation Data Model* (WADM)[14], a vocabulary to express annotations. The basic constructs of the WADM are annotations with at least a target indicating what is described and a body indicating the description.

```
PREFIX oa: <http://www.w3.org/ns/oa#>
<anno>  a              oa:Annotation ;
```

[13] `PREFIX dbo: <http://dbpedia.org/ontology/>`

[14] W3C Candidate Recommendation 2016-09-06, `http://www.w3.org/TR/annotation-model`

```
oa:hasTarget   <target>       ;
oa:hasBody     <body>         .
```

In qa, a question is assumed to be exposed at some URI (e.g. URIQuestion that can be internal and does not need to be public) and is of type qa:Question. Similarly other QA concepts (qa:Answer, qa:Dataset, etc.) are defined in the vocabulary; please see [16] for further details. As a result, when using qa, the knowledge of the QA system is now representable independently from a particular programming language or implementation paradigm because everything is represented as direct or indirect annotations of a resource of type qa:Question. The qa vocabulary is published at the persistent URI https://w3id.org/wdaqua/qanary# under the CC0 1.0 license[15].

The qa vocabulary led to the Qanary methodology [3] for implementing processes operating on top of the knowledge about the question currently processed within a QA system, leading to the possibility of easy-to-reuse QA components. All the knowledge related to questions, answers and intermediate results is stored in a central Knowledge Base (KB). The knowledge is represented in terms of the qa vocabulary in the form of annotations of the relevant parts of the question.

Within Qanary the components all implement the same service interface. Therefore, all components can be integrated into a QA system without manual engineering effort. Via it's service interface a component receives information about the KB (i.e., the endpoint) storing the knowledge about the currently processed question of the user. Hence, the common process within all components is organized as follows:

1. A component fetches the required knowledge via (SPARQL) queries from the KB. In this way, it gains access to all the data required for its particular process.
2. The custom component process is started, computing new insights of the user's question.
3. Finally, the component pushes the results back to the KB (using SPARQL).

Therefore, after each process step (i.e., component interaction), the KB should be enriched with new knowledge (i.e., new annotations of the currently processed user's question). This way the KB keeps track of all the information generated in the QA process even if the QA process is not predefined or not even known. A typical QA pipeline consists of several steps such as NER, NED, relation identification, semantic analysis, query computation and result ranking. Most recently we provided a reference implementation of Qanary [17]. We call this implementation *message-driven*; it follows the architectural pattern that we have previously described for search engines in [4]. The processing steps might be implemented in different components with dedicated technology provided by distinguished research groups. The message-driven implementation of Qanary [4] laid foundations for the QA ecosystem. The advantage of such an ecosystem is that it combines different approaches, functionality, and advances in the QA community under a single umbrella.

[15] https://creativecommons.org/publicdomain/zero/1.0/

4 The Qanary Ecosystem

The Qanary ecosystem consists of a variety of components and services that can be used during a QA process. We describe in the following what components and services are available.
The Qanary ecosystem includes various components covering a broad field tasks within QA systems. This includes different components performing NER like FOX [18] and Stanford NER [8] and components computing NED such as DBpedia Spotlight and AGDISTIS [20]. Also industrial services such as the Alchemy API are part of the ecosystem. Furthermore, Qanary includes a language detection module [13] to identify the language of a textual question. A baseline automatic speech recognition component is also included in the reference implementation. It translates audio input into natural language texts and is based on Kaldi[16]. Additionally it should be noted that a monolithic QA system component was developed in the course of the WDAqua project[17] and is integrated in Qanary. Additional external QA components are included in the ecosystem. In particular, Qanary includes two components from the OKBQA challenge[18] namely the template generation and disambiguation component. All components are implemented following the REST principles. Hence, these tools/approaches become easy to reuse and can now be invoked via transparent interfaces. To make it easy to integrate a new component we have created a Maven archetype that generates a template for a new Qanary component[19]. The main services are encapsulated in the *Qanary Pipeline*. It provides, for example, a service registry. After being started, each component registers itself to the *Qanary Pipeline* central component following the local configuration[20] of the component. Moreover, the *Qanary Pipeline* provides several web interfaces for machine and also human interaction (e.g., for assigning a URI to a textual question, retrieving information about a previous QA process, etc.). Particularly, as each component automatically registers itself to the *Qanary Pipeline*, a new QA system can be created and executed just by on-demand configuration (a concrete one is shown in Fig. 1). Hence, the reference implementation already provides the features required for QA systems using components distributed over the Web.

An additional interface allows for benchmarking a QA system created on demand using Gerbil for QA[21], thus allowing third-party evaluation and citeable URIs. Fig. 2 illustrates the complete reference architecture of Qanary and a few of its components. Additional services include a user interface for a fully working QA system. A running demo can be found at `http://www.wdaqua.eu/qa`.

[16] `http://kaldi-asr.org`
[17] `http://www.wdaqua.eu`
[18] `http://www.okbqa.org/`
[19] `https://github.com/WDAqua/Qanary/wiki/How-do-I-integrate-a-new-component-in-Qanary%3F`
[20] The configuration property `spring.boot.admin.url` defines the endpoint of the central component (and can be injected dynamically).
[21] `http://gerbil-qa.aksw.org`

The code is maintained in the repository at `https://github.com/WDAqua/Qanary` under the MIT License[22].

5 Gaining new insights into the QA process: The EL task

To show how Qanary can be used to gain new insights into QA processes we focus here on the EL task. We present the `qa` vocabulary used to represent the information produced by NER and NED tools. Moreover we describe the components of the Qanary ecosystem that are integrated using the Qanary methodology and that can be used for the EL task. We describe how we constructed a benchmark for EL out of QALD. The analysis of the benchmark will show: what are the best tools to tackle QALD, where are current research gaps, and for which questions do single tools fail and why.

Finally, we present a new dataset that can be used as a gold standard for a sub-task of the QA process.

The following workflow is not restricted to the EL task but can be applied to any other sub-task of the QA process to gain new insights into QA processes.

5.1 The Qanary vocabulary for the EL task

The `qa` vocabulary is designed to be extensible so as not to constrain the creativity of the QA community developers. All information that can possibly be generated and that might need to be shared across QA components can be expressed using new annotations. This principle follows the understanding that standards that allow communication between QA components must be defined by the community. Taking into consideration the state-of-the-art (e.g., [2,20,8]), the `qa` vocabulary was extended with standard concepts for NER and NED representations. This in particular uniforms the representation of the input and output of every integrated component, making it easy to compare and analyze the integrated tools. Note that this does not only hold for tools that can be used for EL but for every tool integrated into the Qanary ecosystem.

To describe an entity spotted within a question we introduced a dedicated annotation:

```
qa:AnnotationOfSpotInstance a owl:Class;
    rdfs:subClassOf qa:AnnotationOfQuestion .
```

If in the question "When was Barack Obama born?" a spotter detects "Barack Obama" as an NE, this fact can be expressed by the following annotation, where `oa:SpecificResource` and `oa:hasSelector` are concepts of the WADM to select a part of a text.

```
<anno1>  a  qa:AnnotationOfSpotInstance .
<anno1>  oa:hasTarget [
    a oa:SpecificResource ;
       oa:hasSource    <URIQuestion>;
       oa:hasSelector  [
```

[22] `https://opensource.org/licenses/MIT`

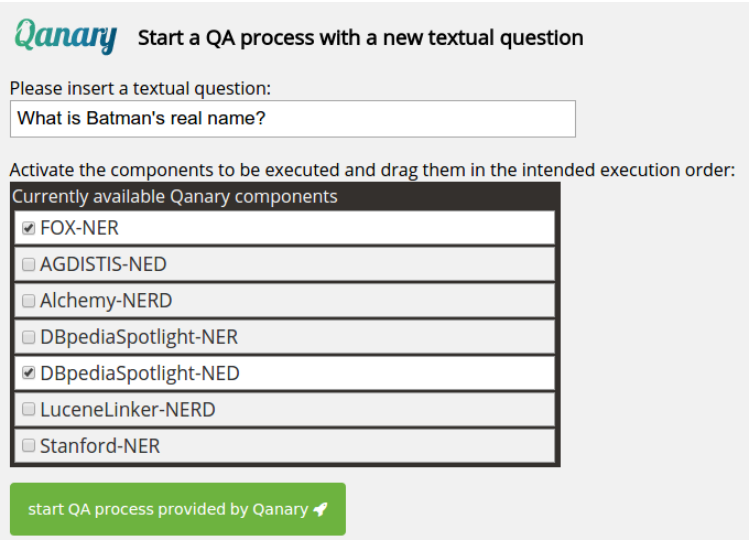

Fig. 1. Snapshot of the Web interface for defining a textual question and a sequence of components to process it (here only NED/NER components where registered).

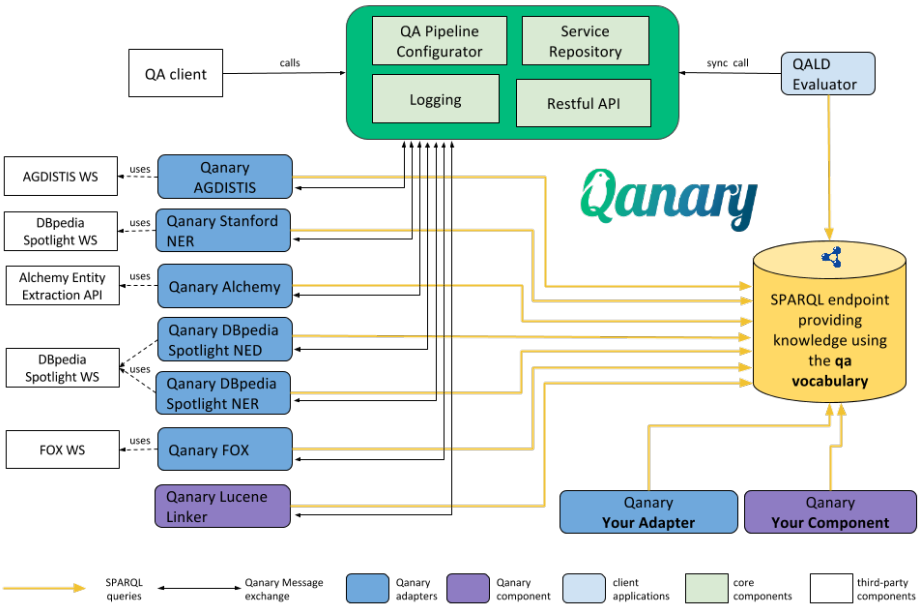

Fig. 2. The Qanary reference architecture implementation highlighting the NER/NED components.

```
        a oa:TextPositionSelector;
        oa:start "9"^^xsd:nonNegativeInteger;
        oa:end   "21"^^xsd:nonNegativeInteger
      ]
] .
```

For named entities, we define the new concept `qa:Named Entity` and a corresponding annotation subclass (i.e., annotations of questions whose body is an instance of `qa:NamedEntity`):

```
qa:NamedEntity          a owl:Class ;
qa:AnnotationOfInstance a owl:Class ;
    owl:equivalentClass [
        a                    owl:Restriction ;
        owl:onProperty       oa:hasBody ;
        owl:someValuesFrom qa:NamedEntity
        ] ;
    rdfs:subClassOf  qa:AnnotationOfQuestion .
```

If an NED tool detects in the question "When was Barack Obama born?" that the text "Barack Obama" refers to `dbr:Barack_Obama`[23], then this can be expressed (using `oa:hasBody`) as:

```
<anno1>  a  qa:AnnotationOfInstance ;
  oa:hasTarget [
    a              oa:SpecificResource ;
    oa:hasSource   <URIQuestion> ;
    oa:hasSelector [
        a        oa:TextPositionSelector ;
        oa:start "9"^^xsd:nonNegativeInteger;
        oa:end   "21"^^xsd:nonNegativeInteger
    ]
  ] ;
  oa:hasBody       <dbr:Barack_Obama> .
```

Note that using annotations provides many benefits thanks to the inclusion of additional metadata such as the creator of the information, the time and a trust score. However, this information is omitted here for readability.

5.2 Reusable NER and NED components

The following components integrated into the Qanary ecosystem solve the task of NER, NED or the combined task of EL.

- **Stanford NER (NER)** is a standard NLP tool that can be used to spot entities for any ontology, but only for languages where a model is available (currently English, German, Spanish and Chinese) [8].
- **FOX (NER)** integrates four different NER tools (including the Stanford NER tool) using ensemble learning [18].
- **DBpedia Spotlight spotter (NER)** uses lexicalizations, i.e., ways to express named entities, that are available directly in DBpedia [12].
- **DBpedia Spotlight disambiguator (NED),** the NED part of DBpedia Spotlight, disambiguates entities by using statistics extracted from Wikipedia texts [12].

[23] `PREFIX dbr: <http://dbpedia.org/resource/>`

- **AGDISTIS (NED)** is a NED tool that uses the graph structure of an ontology to disambiguate entities [20].
- **ALCHEMY (NER + NED):** Alchemy API[24] is a commercial service (owned by IBM) exposing several text analysis tools as web services.
- **Lucene Linker (NER + NED)** is a component that we implemented following the idea of the SINA QA system [15], which employs information retrieval methods.

Note that thanks to the integration as Qanary components all these tools can be interwoven, i.e., each NER tool can be combined with each NED tool just by configuration.

5.3 A QALD-based benchmark for EL in QA

To compare the different entity linking approaches, we created a benchmark based on the QALD (Question Answering over Linked Data) benchmark used for evaluating complete QA systems. The QALD-6 training set[25], which is the recent successor of QALD-5 [19], contains 350 questions, including the questions from previous QALD challenges. For each question, it contains a SPARQL query that retrieves the corresponding answers. For example, the following SPARQL query corresponds to the question "Which soccer players were born on Malta?".

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?uri WHERE {
    ?uri  a               dbo:SoccerPlayer .
    ?uri  dbo:birthPlace  dbr:Malta .
}
```

EL tools should provide functionality to interlink the named entities present in the question with DBpedia (or other data), i.e., they should be able to identify "Malta" and link to it the resource `http://dbpedia.org/resource/Malta`. Our benchmark compares the URIs generated by an EL tool with the resource URIs in the SPARQL query (i.e., those in the `http://dbpedia.org/resource/` namespace)[26], which are obviously required for answering the question. Hence the gold standard for each question is given by all resource URIs in the SPARQL query.[27]

[24] `http://alchemyapi.com`

[25] Training Questions of Task 1: `http://qald.sebastianwalter.org/index.php?x=challenge&q=6`

[26] Note that resources of the type `http://dbpedia.org/ontology/*` would match a DBpedia property or class and therefore are not considered here.

[27] This definition of the gold standard ignores the *order* of the URIs. In practice this definition rarely causes problems, but in theory one could construct counter-examples that pinpoint the limitations. Imagine the question "What German actors were not born in Germany?", and imagine that the word "German" got linked to the entity *dbr:Germany* and "Germany" to *dbr:Germans* – clearly a wrong linking, but "correct" w.r.t. our gold standard definitions. However, in QALD there are no questions in which such a mix-up is likely to happen.

$$\text{Precision}(q) = \frac{\#\text{ correct URIs retrieved by the EL configuration for } q}{\#\text{ URIs retrieved by the EL configuration identifying NE in } q}$$

$$\text{Recall}(q) = \frac{\#\text{ correct URIs retrieved by the EL configuration for } q}{\#\text{ gold standard answers for } q}$$

$$F_1\text{-measure}(q) = 2 \times \frac{\text{Precision}(q) \times \text{Recall}(q)}{\text{Precision}(q) + \text{Recall}(q)}$$

Fig. 3. Metrics used in the EL benchmark

The metrics for a question q are calculated as defined in the QALD benchmark and are reported in Fig. 3[28]. The metrics over all questions are defined as the average of the metrics over the single questions. The corresponding benchmark component is available at `https://github.com/WDAqua/Qanary`.

Note that this procedure can be generalized and applied to many sub-processes of a QA pipeline. For example, one might establish a benchmark to recognize relations or classes, a benchmark to identify the type of the SPARQL query required to implement a question (i.e., a *SELECT* or an *ASK* query), a benchmark for identifying the answer type (i.e., list, single resource, ...) and so on.

We used our benchmarking resource described above to evaluate EL tools. We have identified different strategies to annotate entities in questions. These include using the spotters Stanford NER, FOX, DBpedia Spotlight Spotter, the NED tools AGDISTIS, and the DBpedia Spotlight disambiguator, as well as the (monolithic w.r.t. NER and NED) EL tools Alchemy and Lucene Linker. Each of them is implemented as an independent Qanary component, as presented in Sec. 5.2. According to the Qanary methodology the computed knowledge about a given question is represented in terms of the `qa` vocabulary and can be interpreted by the benchmark component.

For the benchmark all three NER components are combined with each of the two NED components. All questions of QALD-6 are processed by each of the six resulting configurations, and by the two monolithic tools. The benchmark was executed exclusively using the service interface of the *Qanary Pipeline*.

Table 2 shows the benchmark results. The "Fully detected" column indicates the number of questions q where some resources were expected and the EL configuration achieved Recall(q)=1. The column "Correctly Annotated" indicates for how many questions we obtained Precision(q)=Recall(q)=1. Finally, the table shows for each configuration the precision and recall metrics over all questions.

[28] In the corner cases where the number of system answers or the number of gold standard answers is zero we follow the same rules that are used in the QALD evaluation; see `https://github.com/ag-sc/QALD/blob/master/6/scripts/evaluation.rb`.

Pipeline configuration	Fully detected	Correctly Annotated	Precision	Recall	F_1-measure
StanfordNER + AGDISTIS	200	195	0.76	0.59	0.59
StanfordNER + Spotlight disamb.	209	189	0.77	0.62	0.61
FOX + AGDISTIS	189	186	0.83	0.56	0.56
FOX + Spotlight disambiguator	199	192	0.86	0.59	0.58
Spotlight Spotter + AGDISTIS	209	204	0.75	0.62	0.62
Spotlight spotter + disambiguator	242	**213**	0.76	0.71	**0.68**
Lucene Linker	**272**	0	0.01	**0.78**	0.03
Alchemy	143	139	**0.91**	0.42	0.42

Table 2. Benchmark of the QALD-6 data using the Qanary reference implementation.

5.4 Discussion

We presented the Qanary methodology, which allows to interweave the analysed tools. Thanks to the `qa` vocabulary we can collect (from the SPARQL endpoint) the results produced by every configuration. A detailed overview showing for each question if the pipeline configurations lead to a recall resp. F-measure of 1 can be found at:

`https://raw.githubusercontent.com/WDAqua/Qanary/master/ICWE-results/`

- `Recall_1.csv` and
- `F-measure_1.csv`.

We analysed both this data and the results presented in Table 2 to draw some conclusions on the performance of the used tools with respect to QALD.

For some QALD-6 questions none of the pipeline configurations is able to find the required resources, for example:

- *Q1:* "Give me all cosmonauts." with the following resources requested in the SPARQL query: `dbr:Russia`, `dbr:Soviet_Union`. For this question one should be able to understand that cosmonauts are astronauts born either in Russia or in the Soviet Union. Detecting such resources would require a deep understanding of the question. Q201 is similar: "Give me all taikonauts.".
- *Q13:* "Are tree frogs a type of amphibian?"; requested resources: `dbr:Hylidae`, `dbr:Amphibian`.
  The problem here is that the scientific name of "tree frogs" is Hylidae and there is no such information in the ontology except in the free text of the Wikipedia abstract.
- *Q311:* "Who killed John Lennon?"; requested resource: `dbr:Death_of_John_Lennon`.
  The problem is that one would probably assume that the information is encoded in the ontology as a triple like "John Lennon", "killed by", "Mark David Chapman" but this is not the case. Even if in the question the actual

NE is "John Lennon", DBpedia happens to encode the requested information in the resource "Death of John Lennon". A similar case is Q316 ("Which types of grapes grow in Oregon?"), where the resource `dbr:Oregon_wine` is searched.

Spotter comparison An unexpected result is that FOX as a spotter has a lower recall than the Stanford NER tool, even though FOX also includes the results of Stanford NER. This can be seen from comparing the recall of the configurations that combine these tools with AGDISTIS or the DBpedia Spotlight disambiguator. The reason is that, for example, in Q101 ("Which German cities have more than 250,000 inhabitants?") the word "German" is tagged by the Stanford NER tool as "MISC" (miscellaneous). However, FOX only supports the tags "PERS" (person), "ORG" (organisation), and "LOC" (location). This explains why FOX has a lower recall but a higher precision than Stanford NER.

The spotters based on NLP (e.g., Stanford NER and FOX) perform worse than the DBpedia Spotlight Spotter, which is mainly based on vocabulary matching. Syntactic features do not suffice to identify "Prodigy" in Q114 ("Give me all members of Prodigy.") or "proinsulin" in Q12 ("Is proinsulin a protein?"). Moreover, there are cases like Q109 ("Give me a list of all bandleaders that play trumpet."), where bandleaders is not an NE in the NLP sense but is modeled as a resource in the DBpedia *ontology*. Similarly, in Q133 ("Give me all Australian nonprofit organizations."), the resource `dbr:Australia` is expected for the adjective "Australian".

NED comparison The results show that the DBpedia Spotlight disambiguator performs better than AGDISTIS w.r.t. QA. AGDISTIS works on co-occurrences of NE. These occur often in longer texts but are rare in questions. If only one NE is spotted, AGDISTIS can only decide based on the popularity of the resources but not on the context as DBpedia Spotlight does.

EL comparison The best spotter, the DBpedia Spotlight spotter, and the best NED, the DBpedia Spotlight disambiguator, also perform best in the EL task. Only the Lucene Linker has a higher recall but must be followed by a disambiguation tool in the next step to increase precision. The Alchemy API shows the lowest recall.

Our evaluation does not permit the conclusion that the combination of DBpedia Spotlight spotter and disambiguator should be recommended in general. The best choice may depend on the questions and on the particular form of dataset. The DBpedia Spotlight disambiguator, for example, is tightly connected to Wikipedia; even its algorithm cannot be ported to other ontologies. Alchemy, despite showing a very low F_1-score and recall, could be a useful resource for QA over other datasets, such as Freebase or Yago. This is one of the many reasons that makes Qanary in general a valuable resource. For a new QA scenario, Qanary empowers developers to quickly combine existing tools and more easily determine the best configuration. Moreover, a detailed analysis of the configurations can help to detect the main problems of the different strategies to further improve the complete QA process or just individual components. Hence, using

```
PREFIX qa:  <http://www.wdaqua.eu/qa#>
PREFIX oa:  <http://www.w3.org/ns/openannotation/core/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX git: <https://github.com/dbpedia-spotlight/>
<anno1> a              qa:AnnotationOfInstance> ;
        oa:annotatedAt "2016-04-30T15:00:43.687+02:00"^^xsd:dateTime ;
        oa:hasTarget [
                a           oa:SpecificResource> ;
                oa:Selector [
                            a        oa:TextPositionSelector ;
                            oa:end   4 ;
                            oa:start 21
                          ] ;
                oa:hasSource  <URIQuestion>
                ] .
        oa:hasBody     dbr:Margaret_Thatcher ;
        oa:annotatedBy git:dbpedia-spotlight .
```

Fig. 4. Example data of Q178: "Was Margaret Thatcher a chemist?"

Qanary provides insights on the quality of each component w.r.t. the current use case, leading to an optimization of the system based on the given data.

A combination of tools solving the same task in an ensemble learning approach is now possible and is recommended as the benchmark results already indicate. Note that such an analysis is not possible using existing benchmarking tools such as Gerbil or Gerbil for QA since they only provide a final overall score. On the other hand such an analysis is needed to detect existing research gaps and push the advancement of QA further. Hence, following the Qanary methodology the research community is enabled to develop new QA processes and components in a joint engineering effort and to validate the given quality metrics within the specific QA scenario. This again proves the potential impact of Qanary within the engineering process of QA systems.

5.5 Dataset of Annotated Questions for Processing in QA systems

We provide a new dataset with questions of the QALD-6 benchmark, which are completely annotated with disambiguated named entities (DBpedia resource URIs) computed by applying our benchmarking to the EL configurations described in Sec. 5.3. This dataset contains 267 questions (out of 350 questions in the QALD-6 training set) because the components could not annotate the rest. A Turtle file, representing the results in terms of the `qa` vocabulary, is published at `https://github.com/WDAqua/Qanary` under the CC0 license. A typical fragment of the provided data is provided in Fig. 4.

We imagine the following usage scenarios for this dataset. It can be used as input for steps in a QA process *following* the EL task that require annotated named entities, such as relation detection or query computation. Consequently, QA components that depend on the output of the EL task can now be evaluated without depending on concrete EL components (and without the results being

influenced by possible flaws). Hence, in conjunction with the SPARQL queries already defined in QALD-6, we established a new gold standard for evaluating parts of a QA process. We also provide the component for computing this dataset (cf., Sec. 5.3); it can be extended if improved EL configurations are available or when a new version of the QALD benchmark is released.

6 Conclusion and Future Work

We have presented the status of the Qanary ecosystem, which includes a variety of components and services that can be used by the research community. These include typical components for sub-tasks of a QA pipeline as well as a number of related services.

Since all messages exchanged between components are expressed using the `qa` vocabulary, all information generated during a QA process can be easily kept. Thanks to this uniform message format it is now possible to easily compare existing tools and integrate new ones. Moreover, the Qanary methodology allows to integrate independent, distributed components, implemented in different programming languages in a loosely-coupled fashion. This allows the creation of comprehensive QA systems in a community effort.

Driven by the demand for better QA technology, we propose a general workflow to develop future QA systems. It mainly breaks down into two parts: (1.) the identification and integration of existing state-of-the-art approaches to solve a particular sub-task in the QA pipeline, and (2.) the derivation of a benchmark from benchmarks for QA such as QALD. Additionally a new gold standard for the sub-task can be provided. In contrast to other approaches the `qa` vocabulary allows to analyse a QA process. Hence, full traceability of the information used in the QA process is ensured, enabling, for example, the optimization of the assigned components. Additionally, the Qanary methodology allows to create such processes in a flexible way. This allows researchers to focus on particular tasks taking advantage of the results of the research community and contributing to it directly in a reusable way.

We have demonstrated this workflow in the case of EL. This way we realized a set of reusable components as well as the first benchmark for EL in the context of QA. All together we have shown how Qanary can be used to gain deep insights in QA processes. While having such insights the engineering process can be steered efficiently towards the improvement of the QA components. Hence, the presented engineering approach is particularly well suited for experimental and innovation-driven approaches (e.g., used by research communities).

The Qanary ecosystem is maintained and used within the WDAqua ITN project[29] (2015–2018 [10]), where Qanary is the reference architecture for new components. All artifacts are published under permissive open licenses: MIT for the software, CC0 for the datasets and the vocabulary.

One of our future task is to populate the Qanary ecosystem with any component significant to the QA community. According to the literature, the tasks of

[29] `http://wdaqua.eu`

relation and answer type detection are of particular relevance, but not yet sufficiently covered by existing components. Additionally, as Qanary provides easy access to different implementations having the same purpose, ensemble learning components for all steps within a QA process are becoming possible and will increase the flexibility as well as boost overall QA quality. Hence, our overall goal is to provide a fully-featured ecosystem for creating QA components and concurrently supporting the measuring and improvement of particular QA systems w.r.t. the considered use cases. This aim provides several research challenges, e.g., the (semi-)automatic creation of self-optimizing QA systems.

Acknowledgments Parts of this work received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 642795, project: Answering Questions using Web Data (WDAqua). We would like to thank Elena Demidova for proof-reading.

References

1. S. Atdag and V. Labatut. A comparison of named entity recognition tools applied to biographical texts. In *2nd International Conference on Systems and Computer Science (ICSCS)*, 2013.
2. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. 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. Proceedings*, ISWC'07/ASWC'07. Springer, 2007.
3. A. Both, D. Diefenbach, K. Singh, S. Shekarpour, D. Cherix, and C. Lange. Qanary – a methodology for vocabulary-driven open question answering systems. In *The Semantic Web. Latest Advances and New Domains: 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 – June 2, 2016, Proceedings*, 2016.
4. A. Both, A.-C. Ngonga Ngomo, R. Usbeck, D. Lukovnikov, Ch. Lemke, and M. Speicher. A service-oriented search framework for full text, geospatial and semantic search. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 65–72. ACM, 2014.
5. E. Cabrio, J. Cojan, A. P. Aprosio, B. Magnini, A. Lavelli, and F. Gandon. QAKiS: an open domain QA system based on relational patterns. In Birte Glimm and David Huynh, editors, *Proc. of the ISWC 2012 Posters & Demonstrations Track*, volume 914 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
6. C. Dima. Answering natural language questions with intui3. In *CLEF (Working Notes)*, 2014.
7. Ó. Ferrández, Ch. Spurk, M. Kouylekov, I. Dornescu, S. Ferrández, M. Negri, R. Izquierdo, D. Tomás, C. Orasan, G. Neumann, B. Magnini, and J.L.V. González. The QALL-ME framework: A specifiable-domain multilingual Question Answering architecture. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2), 2011.
8. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, 2005.
9. A. Freitas, J. Oliveira, E. Curry, S. O'Riain, and J. da Silva. Treo: combining entity-search, spreading activation and semantic relatedness for querying linked data. In *1st Workshop on Question Answering over Linked Data (QALD-1)*, 2011.

10. Ioanna Lytra, Maria-Esther Vidal, Christoph Lange, Sören Auer, and Elena Demidova. WDAqua – answering questions using web data. In Erik Mannens, Mauro Dragoni, Lyndon Nixon, and Oscar Corcho, editors, *EU Project Networking*, 2016.
11. E. Marx, R. Usbeck, A. Ngonga Ngomo, K. Höffner, J. Lehmann, and S. Auer. Towards an open question answering architecture. In *10th International Conference on Semantic Systems*, 2014.
12. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, 2011.
13. Shuyo Nakatani. Language detection library for java, 2010. https://github.com/shuyo/language-detection.
14. G. Rizzo and R. Troncy. NERD: A framework for unifying named entity recognition and disambiguation extraction tools. In *13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012.
15. S. Shekarpour, E. Marx, A.-C. Ngonga Ngomo, and S. Auer. SINA: Semantic interpretation of user queries for question answering on interlinked data. *Web Semantics: Science, Services and Agents on the WWW*, 30, 2015.
16. K. Singh, A. Both, D. Diefenbach, and S. Shekarpour. Towards a message-driven vocabulary for promoting the interoperability of question answering systems. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, 2016.
17. K. Singh, A. Both, D. Diefenbach, S. Shekarpour, D. Cherix, and C. Lange. Qanary–the fast track to create a question answering system with linked data technology. In *The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016, Revised Selected Papers*, 2016.
18. R. Speck and A. Ngonga Ngomo. Ensemble learning for named entity recognition. In *The Semantic Web – ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I.* Springer International Publishing, 2014.
19. C. Unger, C. Forascu, V. Lopez, A. Ngonga Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question answering over linked data (QALD-5). In *CLEF (Working Notes)*, 2015.
20. R. Usbeck, A. Ngonga Ngomo, M. Röder, D. Gerber, S. Coelho, S. Auer, and A. 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.* Springer, 2014.
21. R. Usbeck, M. Röder, A. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, P. Ferragina, C. Lemke, A. Moro, R. Navigli, F. Piccinno, G. Rizzo, H. Sack, R. Speck, R. Troncy, J. Waitelonis, and L. Wesemann. GERBIL: General entity annotator benchmarking framework. In *24th International Conference on World Wide Web*, 2015.