WDAqua-core1: A Question Answering service for RDF Knowledge Bases

Dennis Diefenbach
Université de Lyon, CNRS UMR 5516
Laboratoire Hubert Curien
Saint-Etienne, France
dennis.diefenbach@univ-st-etienne.fr

Kamal Singh
Université de Lyon, CNRS UMR 5516
Laboratoire Hubert Curien
Saint-Etienne, France
kamal.singh@univ-st-etienne.fr

Pierre Maret
Université de Lyon, CNRS UMR 5516
Laboratoire Hubert Curien
Saint-Etienne, France
pierre.maret@univ-st-etienne.fr

ABSTRACT

In the last two decades a new part of the web grew significantly, namely the Semantic Web. It contains many Knowledge Bases (KB) about different areas like music, books, publications, live science and many more. Question Answering (QA) over KBs is seen as the most promising approach to bring this data to end-users. We describe WDAqua-core1, a QA service for querying RDF knowledge-bases. It is multilingual, it supports different RDF knowledge bases and it understands both full natural language questions and keyword questions.

KEYWORDS

Question Answering over Knowledge Bases, QALD, Mutlilinguality, Robustness, Portability

ACM Reference Format:
Dennis Diefenbach, Kamal Singh, and Pierre Maret. 2018. WDAqua-core1: A Question Answering service for RDF Knowledge Bases. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3184558.3191541

1 INTRODUCTION

Question Answering (QA) over Knowledge Bases (KB) is a research topic that emerged in the last two decades. We want to explain the problem using an example.

In a KB the information is stored in triples. The information that "Saint-Étienne is in the department of the Loire" is stored in DBpedia[1] (a well known KB) as one triple:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

dbr:Saint-Étienne dbo:department dbr:Loire .
```

[1]http://wiki.dbpedia.org

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191541

Imagine a user asks: "In which department is Saint-Étienne?". We want to directly find the answer for the question, i.e. we want to retrieve the information stored in the triple above. This is generally done by converting the question into a SPARQL query. For the above question we would like to generate the following SPARQL query:

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?x WHERE {
    dbr:Saint-Étienne dbo:department ?x
}
```

This will retrieve the desired answer.
The impact of a QA solution that has good performance and that is easily adaptable to new domains is huge. Every institution, company or even private people can publish data in RDF and allow users to find this data easily using natural language. The first industrial applications are already used in our everyday life. These include *Apple Siri*[2], *Microsoft Cortana*[3], and *Ok Google*[4]. Unfortunately these systems query only some internal KBs. Having a QA solution freely available would allow to set up QA services over many new data sources and would probably also boost the publication of new RDF data-sets. We present one of the few available services for QA over KB and describe its main features.
This publication is organized as follows. In section 2, we describe related work, in section 3, we describe the proposed approach and in section 4, we describe the integration of the QA system with existing services. We present the evaluation of our approach in section 5, and we conclude with section 6.

2 RELATED WORK AND RESEARCH PROBLEM

Question Answering over KBs is a very hot research topic. One can count more than 60 publications about QA systems over KB and many more in related topics. A complete overview of the published approaches can be found in [9].
Some drawbacks that current proposed approaches have are: most systems are limited to one language namely English, QA systems are generally claimed to be open domain, but they are generally only applied to one KB and, most systems were only tested for

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

well-formulated natural language questions and were not evaluated over keyword questions. There are mainly two reasons.
The first is that many QA systems rely on large data sets to be trained. These are very expensive to create. For example Berant et al. [3] report that they spent several thousands dollars for the creation of WebQuestions (one of the most popular benchmarks for QA) using Amazon Mechanical Turk. Since it is difficult, time-consuming and expensive to create new QA data-sets it is also difficult to move to new languages and knowledge bases. This is for example the case of Bordes et al. [4] and Zhang et al. [26].
A second reason is that many QA systems rely on hard coded rules on top of NLP tools. These rules are often adapted to a specific KB and are clearly language dependent. This is for example the case of Xser [24], gAnswer [28] or QuerioDali[17].
There are very few systems that are running as web-services. A few exceptions are Platypus (https://askplatyp.us/), Qakis (http://qakis.org) and gAnswer2 (http://ganswer.gstore-pku.com).

3 PROPOSED APPROACH

We propose a new approach to construct a SPARQL query from given questions. Imagine the QA system receives the question: "Give me actors born in Strasbourg". While a common approach is to first analyze the phrase syntactically and find out that Strasbourg is a named entity, born is a predicate and use these information to create the corresponding SPARQL query. We follow a completely different approach. This approach is based on the idea that many questions can be understood by ignoring the syntax and that the semantics of the words suffice.
It can be decomposed in 4 steps: Query Expansion, Query Construction, Query Ranking and Answer Decision. We want to briefly illustrate the idea using the above example of "Give me actors born in Strasbourg". For further details we refer to [8].
In the **Query Expansion** phase we first see to which possible concepts the consequent words in the question can refer to. For example we take into consideration all meaning that the word "actor" can have according to the underlying KB, i.e. it can refer to a "2003 film by Conor McPherson", a "programming language called actor" or to the concept of actor as it is intended in the question. The word "born" can correspond to the indented meaning in the question, i.e. "the birth place" but could also be a "family name", a "district of the city Brüggen" and many other things. The question is now between all these meanings, which is the right one. To make this decision we look into the KB and we search for the most probable meaning. We interpret this as the identified concepts that are nearest in the graph.
In the **Query Construction** phase, based on the distance of the different concepts in the graph, we construct all possible SPARQL queries. If we apply this strategy to the above question, using as a KB dbpedia, we get over 200 possible SPARQL queries. We call each of this query an interpretation of the questions. Translated in natural language some of these SPARQL queries are just "Give me information about the district of Brüggen", "Give me information about the 2003 film by Conor McPherson." others are "Give me actors related to Strasbourg." and also the searched interpretation of "Give me actors born in Strasbourg.". Note that one will probably not find any connection in the KB between the "2003 film by Conor McPherson" and the "district of Brüggen" so that these concepts are probably not the intended.
In the **Query Ranking** phase, we rank the generated SPARQL queries. We use few features for the linking like: how many words in the questions are associated with resources in the SPARQL queries and how similar is the label of the resource to the corresponding resource in the question. While the ranking can be performed in different ways we use a linear combination of the above features. This means that very few training data is needed to train the ranker. Moreover if no training data is available we just reuse the coefficients learned over a different data-set.
If everything goes right, the first ranked query is the SPARQL query we wanted to generate. Note that we nearly always generate a SPARQL query, in particular also for questions that cannot be answered using the corresponding Knowledge Bases. So optionally we perform the last phase, the **Answer Decision**. For this step some training data is needed. This phase consists of a model that decides, based on the features above, if the top-ranked SPARQL query is an answer or not.

While at first glance, the approach may look quite complex and not so promising, it turns out that it works well and it also has a number of advantages over traditional methods. These include robustness with respect to the asked question, i.e. if the user asks "Give me actors born in Strasbourg", "actors born in Strasbourg", "actors Strasbourg born in", or "born actor Strasbourg" we can always identify the intended meaning. In particular this has as an effect that the approach also works for keyword and syntactically wrong questions. Moreover the approach is multilingual. The same arguments above work also for the German question "Gib mir Schauspieler die in Straßbourg geboren sind.". It turns out that the approach works well even with no to very little training data. This means that one does not need to construct big training data-sets which can be very expensive. Moreover the approach is quite flexible to how the knowledge is encoded into the knowledge base. For example the information that "the capital of Europe is Brussels" can be encoded as the triples:

```
"Brussels" "capital" "Europe"
              OR
"Europe" "capital" "Brussels"
              OR
"Europe" "type" "capital of Europe"
              OR
"Brussels" "type" "capital"
"Brussels" "located in" "Europe"
```

Independently on how the information is encoded, we are able to adapt to the different ways. The need of little training data and the ability to adapt on how the knowledge is encoded allows to easily port the approach to new KBs. We call our system WDAqua-core1. It currently can query 4 different KBs (DBpedia, Wikidata, MusicBrainz and DBLP) and 5 different languages (English, French, German, Italian and Spanish).
While we are completely ignoring the syntax, the syntactic features can be included. For example the syntax is important to catch the difference between: "What is a continent?" and "Give me all continents.". The plural "s" is important in this case. We want to

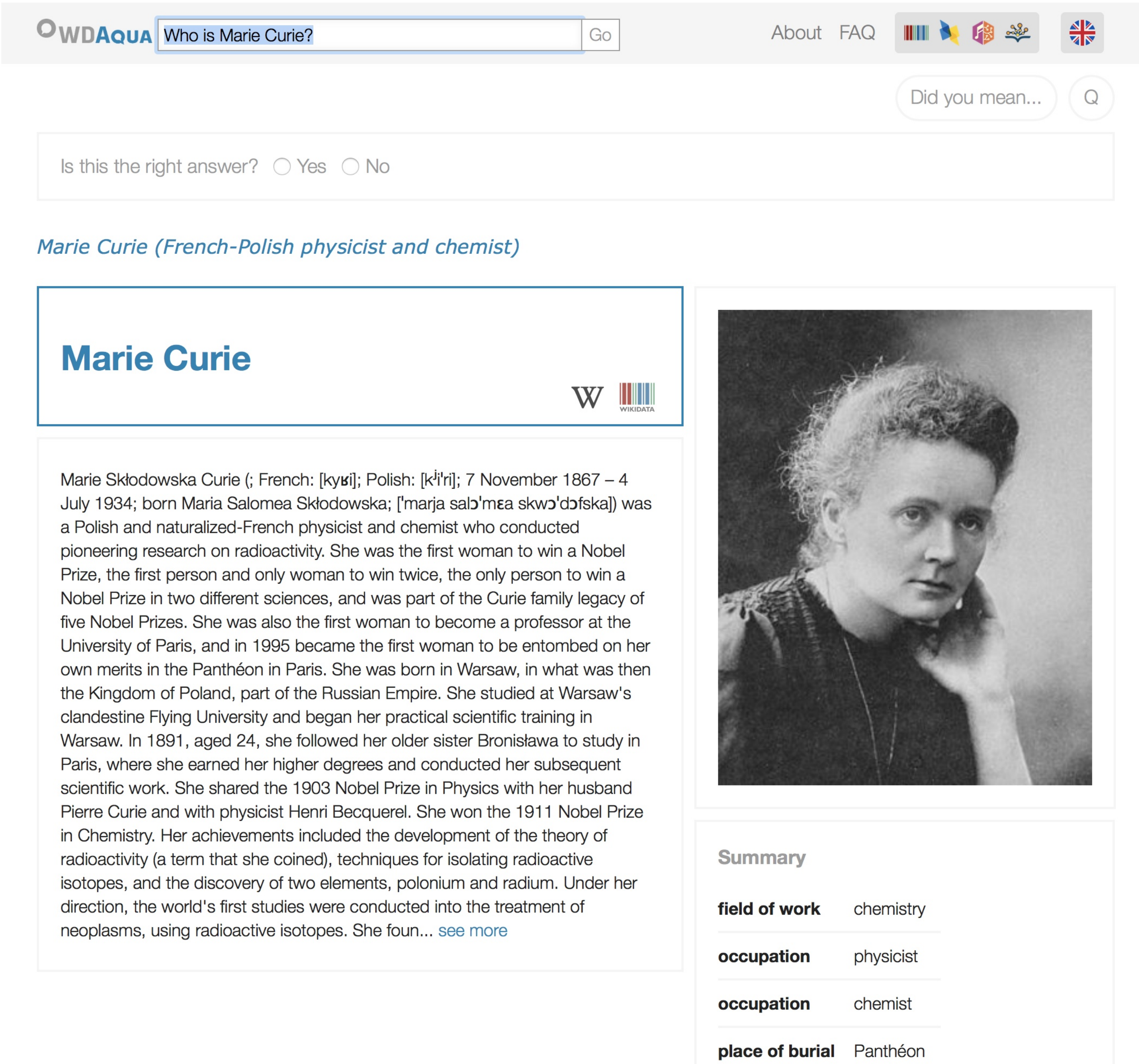

Figure 1: Screen-shot of Trill for the question "Who is Marie Curie?". The following information are presented to the user. A line indicating what the QA system understood. In this case the QA systems understood "Marie Curie (French-Polish physician and chemist)". Trill presents the label of the answer entity "Marie Curie", together with external links to Wikipedia and Wikidata. Furthermore there is description of the answer entity and an image. Finally on the bottom right side there is a summary for the answer entity, i.e. some relevant key facts about it. The summary is generated by the SummaServer [10]

use the techniques used in existing QA systems to catch these differences.

4 INTEGRATION IN QANARY

The approach is integrated as a component into Qanary [5][7], a modular architecture for QA systems. This allows to reuse the QA system and to access it via REST-full interfaces. These allow to interact with the QA system in a programmatical way. For example it offers a REST-full interface to the Gerbil for QA platform [22]. This allows an easy and independent evaluation of the presented service and assures an easy reproducibility of the results.
Moreover it allows to combine it together with existing components

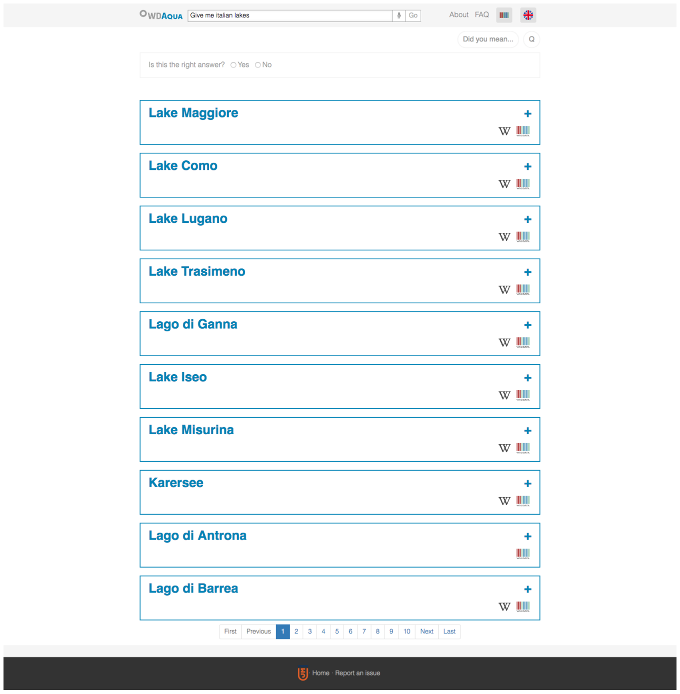

Figure 2: Screen-shot of Trill for the question "Give me italian lakes.". Trill presents the answer as a list. The list is ranked by PageRank scores of the corresponding entities. By clicking on the corresponding entry more information are presented.

like the language detection component and the speech recognition component. Moreover it can be combined with the reusable front-end called Trill [6]. A screen shot of the Trill front-end using WDAqua-core1 in the back-end can be seen in Figure 1 and Figure 2. An online version can be found under:

www.wdaqua.eu/qa.

5 EVALUATION

We evaluated the approach over the QALD benchmarks over both full natural language questions and keyword questions (indicated as "key" and "full"). Moreover, we evaluated the approach over 5 different languages, namely English, French, German, Italian and Spanish (indicated as "en, "fr", "de", "it", "es"). All the results together with the corresponding state-of-the-art systems are reported in Table 1.

Note that the proposed approach outperforms every proposed approach except of Xser [24] and UTQA [23]. These systems required additional training data, than the training data offered by the QALD benchmarks. Moreover, they were not tested over keyword question so that it is not clear how they behave in these situations. Finally these systems are neither available as web-services nor the code is freely available and, thus, their results are not reproducible.

QA system	Lang	Type	P	R	F	Time
QALD-3						
WDAqua-core1	**en**	**full**	**0.58**	**0.46**	**0.51**	**1.08s**
WDAqua-core1	**en**	**key**	**0.65**	**0.44**	**0.52**	**0.81s**
WDAqua-core1	**de**	**full**	**0.88**	**0.29**	**0.44**	**0.22s**
WDAqua-core1	**de**	**key**	**0.92**	**0.29**	**0.44**	**0.18s**
WDAqua-core1	**fr**	**full**	**0.86**	**0.28**	**0.43**	**0.31s**
WDAqua-core1	**fr**	**key**	**0.92**	**0.29**	**0.44**	**0.20s**
WDAqua-core1	**it**	**full**	**0.89**	**0.29**	**0.44**	**0.20s**
WDAqua-core1	**it**	**key**	**0.92**	**0.29**	**0.44**	**0.16s**
WDAqua-core1	**es**	**full**	**0.90**	**0.29**	**0.44**	**0.18s**
WDAqua-core1	**es**	**key**	**0.91**	**0.29**	**0.44**	**0.17s**
gAnswer [28]∗	en	full	0.40	0.40	0.40	≈ 1 s
RTV [14]	en	full	0.32	0.34	0.33	-
Intui2 [11]	en	full	0.32	0.32	0.32	-
SINA [21]∗	en	full	0.32	0.32	0.32	≈10-20s
DEANNA [25]∗	en	full	0.21	0.21	0.21	≈1-50s
SWIP [19]	en	full	0.16	0.17	0.17	-
Zhu et al. [27]∗	en	full	0.38	0.42	0.38	-
QALD-4						
Xser [24]	en	full	0.72	0.71	0.72	-
WDAqua-core1	**en**	**key**	**0.76**	**0.40**	**0.52**	**0.32s**
WDAqua-core1	**en**	**full**	**0.56**	**0.30**	**0.39**	**0.46s**
gAnswer [28]	en	full	0.37	0.37	0.37	0.973 s
CASIA [16]	en	full	0.32	0.40	0.36	-
WDAqua-core1	**de**	**key**	**0.92**	**0.20**	**0.33**	**0.04s**
WDAqua-core1	**fr**	**key**	**0.92**	**0.20**	**0.33**	**0.06s**
WDAqua-core1	**it**	**key**	**0.92**	**0.20**	**0.33**	**0.04s**
WDAqua-core1	**es**	**key**	**0.92**	**0.20**	**0.33**	**0.05s**
WDAqua-core1	**de**	**full**	**0.90**	**0.20**	**0.32**	**0.06s**
WDAqua-core1	**it**	**full**	**0.92**	**0.20**	**0.32**	**0.16s**
WDAqua-core1	**es**	**full**	**0.90**	**0.20**	**0.32**	**0.06s**
WDAqua-core1	**fr**	**full**	**0.86**	**0.18**	**0.29**	**0.09s**
Intui3 [12]	en	full	0.23	0.25	0.24	-
ISOFT [18]	en	full	0.21	0.26	0.23	-
Hakimov [15]∗	en	full	0.52	0.13	0.21	-
QALD-5						
Xser [24]	en	full	0.74	0.72	0.73	-
WDAqua-core1	**en**	**full**	**0.56**	**0.41**	**0.47**	**0.62s**
WDAqua-core1	**en**	**key**	**0.60**	**0.27**	**0.37**	**0.50s**
AskNow[13]	en	full	0.32	0.34	0.33	-
QAnswer[20]	en	full	0.34	0.26	0.29	-
WDAqua-core1	**de**	**full**	**0.92**	**0.16**	**0.28**	**0.20s**
WDAqua-core1	**de**	**key**	**0.90**	**0.16**	**0.28**	**0.19s**
WDAqua-core1	**fr**	**full**	**0.90**	**0.16**	**0.28**	**0.19s**
WDAqua-core1	**fr**	**key**	**0.90**	**0.16**	**0.28**	**0.18s**
WDAqua-core1	**it**	**full**	**0.88**	**0.18**	**0.30**	**0.20s**
WDAqua-core1	**it**	**key**	**0.90**	**0.16**	**0.28**	**0.18s**
WDAqua-core1	**es**	**full**	**0.88**	**0.14**	**0.25**	**0.20s**
WDAqua-core1	**es**	**key**	**0.90**	**0.14**	**0.25**	**0.20s**
SemGraphQA[2]	en	full	0.19	0.20	0.20	-
YodaQA[1]	en	full	0.18	0.17	0.18	-
QuerioDali[17]	en	full		0.48	?	?

QA system	Lang	Type	P	R	F	Time
QALD-6						
UTQA [23]	en	full	0.82	0.69	0.75	-
UTQA [23]	es	full	0.76	0.62	0.68	-
UTQA [23]	fs	full	0.70	0.61	0.65	-
WDAqua-core1	**en**	**full**	**0.62**	**0.40**	**0.49**	**0.93s**
WDAqua-core1	**en**	**key**	**0.52**	**0.32**	**0.40**	**0.68s**
SemGraphQA [2]	en	full	0.70	0.25	0.37	-
WDAqua-core1	**de**	**full**	**0.95**	**0.17**	**0.29**	**0.12s**
WDAqua-core1	**de**	**key**	**0.96**	**0.17**	**0.29**	**0.08s**
WDAqua-core1	**fr**	**full**	**0.91**	**0.16**	**0.27**	**0.37s**
WDAqua-core1	**fr**	**key**	**0.96**	**0.17**	**0.29**	**0.12s**
WDAqua-core1	**it**	**full**	**0.96**	**0.17**	**0.29**	**0.16s**
WDAqua-core1	**it**	**key**	**0.96**	**0.17**	**0.29**	**0.07s**
WDAqua-core1	**es**	**full**	**0.96**	**0.17**	**0.29**	**0.19s**
WDAqua-core1	**es**	**key**	**0.96**	**0.17**	**0.29**	**0.19s**
QALD-7						
WDAqua-core1	**en**	**full**	**0.63**	**0.32**	**0.42**	**0.47s**
WDAqua-core1	**en**	**key**	**0.58**	**0.16**	**0.25**	**0.34s**
WDAqua-core1	**de**	**full**	**0.82**	**0.10**	**0.18**	**0.07s**
WDAqua-core1	**de**	**key**	**0.80**	**0.10**	**0.18**	**0.05s**
WDAqua-core1	**fr**	**full**	**0.80**	**0.10**	**0.18**	**0.17s**
WDAqua-core1	**fr**	**key**	**0.80**	**0.10**	**0.18**	**0.06s**
WDAqua-core1	**it**	**full**	**0.82**	**0.10**	**0.18**	**0.05s**
WDAqua-core1	**it**	**key**	**0.80**	**0.10**	**0.18**	**0.04s**
WDAqua-core1	**es**	**full**	**0.84**	**0.10**	**0.18**	**0.03s**
WDAqua-core1	**es**	**key**	**0.84**	**0.10**	**0.18**	**0.03s**

Table 1: This table summarizes the results obtained by the QA systems evaluated over with QALD-3 (over DB- pedia 3.8), QALD-4 (over DBpedia 3.9), QALD-5 (over DBpedia 2014), QALD-6 (over DBpedia 2015-10), QALD-7 (over DBpedia 2016-04). To find the systems, we used Google scholar to select all publications about QA systems that cited one of the QALD challenge publications. We indicate the average running times of a query for the systems where we found them. Even if the run time evaluations were executed on different hardware, it still helps to give an idea about the scalability.

6 CONCLUSION AND FUTURE WORK

The presented QA approach is novel in different aspects compared to the state-of-the-art. The approach does not focus on the syntax but on the semantics to understand a question. This has mainly two advantages: it works for both keyword and full natural language questions and it is multilingual. In future we want to apply the approach to new knowledge-bases and query multiple knowledge-bases at the same time. The aim is to present a first prototype for a QA system that can potentially query the Semantic Web.

Note: There is a Patent Pending for the presented approach. It was submitted the 18 January 2018 at the EPO and has the number EP18305035.0.

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).

REFERENCES

[1] Baudiš, P., Šedivỳ, J.: QALD challenge and the YodaQA system: Prototype notes

[2] Beaumont, R., Grau, B., Ligozat, A.L.: SemGraphQA@QALD-5: LIMSI participation at QALD-5@CLEF. CLEF (2015)

[3] Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic Parsing on Freebase from Question-Answer Pairs. In: EMNLP (2013)

[4] Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)

[5] Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., Lange, C.: Qanary – a methodology for vocabulary-driven open question answering systems. In: ESWC 2016 (2016)

[6] Diefenbach, D., Amjad, S., Both, A., Singh, K., Maret, P.: Trill: A reusable front-end for qa systems. In: ESWC P&D (2017)

[7] Diefenbach, D., Singh, K., Both, A., Cherix, D., Lange, C., Auer, S.: The Qanary Ecosystem: getting new insights by composing Question Answering pipelines. In: ICWE (2017)

[8] Diefenbach, D., Both, A., Singh, K., Maret, P.: Towards a question answering system over the semantic web (2018), arXiv:1803.00832

[9] Diefenbach, D., Lopez, V., Singh, K., Pierre, M.: Core Techniques of Question Answering Systems over Knowledge Bases: a Survey. Knowledge and Information systems (2017)

[10] Diefenbach, D., Thalhammer, A.: Pagerank and generic entity summarization for rdf knowledge bases. In: ESWC (under review). Springer (2018)

[11] Dima, C.: Intui2: A prototype system for question answering over linked data. Proceedings of the Question Answering over Linked Data lab (QALD-3) at CLEF (2013)

[12] Dima, C.: Answering natural language questions with Intui3. In: Conference and Labs of the Evaluation Forum (CLEF) (2014)

[13] Dubey, M., Dasgupta, S., Sharma, A., Höffner, K., Lehmann, J.: AskNow: A Framework for Natural Language Query Formalization in SPARQL. In: International Semantic Web Conference. Springer (2016)

[14] Giannone, C., Bellomaria, V., Basili, R.: A HMM-based approach to question answering against linked data. Proceedings of the Question Answering over Linked Data lab (QALD-3) at CLEF (2013)

[15] Hakimov, S., Unger, C., Walter, S., Cimiano, P.: Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In: Natural Language Processing and Information Systems. Springer (2015)

[16] He, S., Zhang, Y., Liu, K., Zhao, J.: CASIA@ V2: A MLN-based Question Answering System over Linked Data. Proc. of QALD-4 (2014)

[17] Lopez, V., Tommasi, P., Kotoulas, S., Wu, J.: Queriodali: Question answering over dynamic and linked knowledge graphs. In: International Semantic Web Conference. pp. 363–382. Springer (2016)

[18] Park, S., Shim, H., Lee, G.G.: ISOFT at QALD-4: Semantic similarity-based question answering system over linked data. In: CLEF (2014)

[19] Pradel, C., Haemmerlé, O., Hernandez, N.: A semantic web interface using patterns: the SWIP system. In: Graph Structures for Knowledge Representation and Reasoning. Springer (2012)

[20] Ruseti, S., Mirea, A., Rebedea, T., Trausan-Matu, S.: QAnswer-Enhanced Entity Matching for Question Answering over Linked Data. CLEF (2015)

[21] Shekarpour, S., Marx, E., Ngomo, A.C.N., Auer, S.: Sina: Semantic interpretation of user queries for question answering on interlinked data. Web Semantics: Science, Services and Agents on the World Wide Web 30 (2015)

[22] Usbeck, R., Röder, M., Hoffmann, M., Conrads, F., Huthmann, J., Ngonga-Ngomo, A.C., Demmler, C., Unger, C.: Benchmarking question answering systems. Semantic Web Journal (2016), *to appear*

[23] Pouran-ebn veyseh, A.: Cross-Lingual Question Answering Using Profile HMM & Unified Semantic Space. In: ESWC (2016), *to appear*

[24] Xu, K., Feng, Y., Zhao, D.: Xser@ QALD-4: Answering Natural Language Questions via Phrasal Semantic Parsing (2014)

[25] Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., Weikum, G.: Natural language questions for the web of data. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics (2012)

[26] Zhang, Y., Liu, K., He, S., Ji, G., Liu, Z., Wu, H., Zhao, J.: Question answering over knowledge base with neural attention combining global knowledge information. arXiv preprint arXiv:1606.00979 (2016)

[27] Zhu, C., Ren, K., Liu, X., Wang, H., Tian, Y., Yu, Y.: A Graph Traversal Based Approach to Answer Non-Aggregation Questions Over DBpedia. arXiv preprint arXiv:1510.04780 (2015)

[28] Zou, L., Huang, R., Wang, H., Yu, J.X., He, W., Zhao, D.: Natural language question answering over RDF: a graph data driven approach. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM (2014)