A Question Answering System for retrieving German COVID-19 data-driven and quality-controlled by Semantic Technology

Andreas Both[1], Aleksandr Perevalov[1], Johannes Richard Bartsch[1], Paul Heinze[1], Rostislav Iudin[1], Johannes Rudolf Herkner[1], Tim Schrader[1], Jonas Wunsch[1], Ann Kristin Falkenhain[2], and René Gürth[3]

[1] Anhalt University of Applied Sciences, Köthen (Anhalt), Germany
[2] Federal Ministry of the Interior, Building and Community, Germany
[3] Informationstechnikzentrum Bund (ITZBund), Germany

Abstract. The COVID-19 pandemic is a showcase for a data-driven society. However, making the corresponding data available is not easy due to local characteristics and time-depending metrics. In this paper, we present the Coronabot that is facilitating the access to German Coronavirus pandemic data. It is capable of answering German and English questions. The component-based system understands time-related questions and questions related to the localities of all administrative levels in Germany. The system is driven by semantic technologies from two perspectives. First, for all internal component interaction, RDF is used which is establishing an extensible architecture as well as microbenchmarking the integrated components as quality assurance is a crucial issue w.r.t. health and freedom-restricting scenarios. Second, several components take advantage of data from the Linked Open Data (LOD) cloud.

Keywords: Question Answering · Coronavirus · COVID-19

1 Introduction

During the Coronavirus pandemic, numbers are presented to people on a daily basis. For example, news portals provide such data using texts and interactive maps. Many areas of interest for citizens (e.g., curfews) depend on the concrete administrative regions. In Germany the Coronavirus data is captured and evaluated on the district level (groups of villages and towns) which is an uncommon abstraction level for ordinary citizens as they typically expect an answer for their hometown. Moreover, historic data from weeks or months ago is not easily accessible. Despite other aspects, this might lead to a misunderstanding and misinterpretation of the current situation by politicians, decision makers, and citizens as it is very hard for humans to remember exact numbers for a long period of time. Additionally, debunking wide-spreading misinformation sometimes

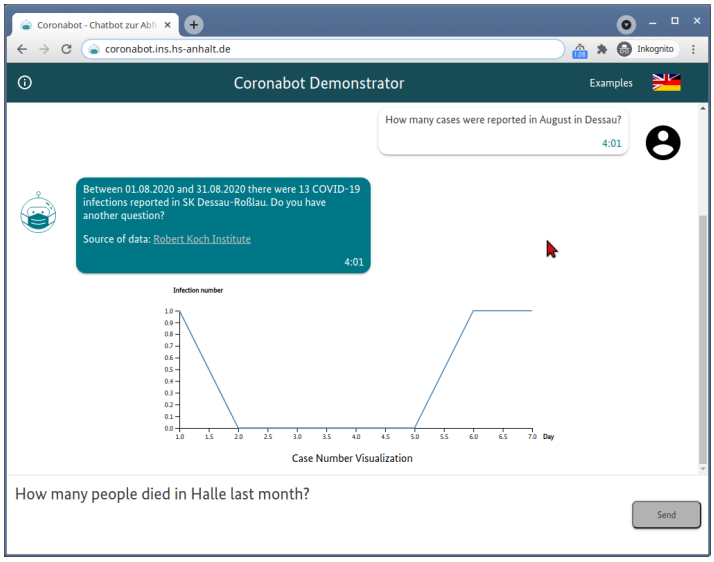

Fig. 1: Coronabot screenshot.

consumes precious resources. Hence, among other major challenges for societies, a pandemic is also a reference example of the need for good data accessibility providing the potential to increase transparency and trust.

Consequently, offering natural-language interfaces (i.e., Question Answering systems, short: QA systems) to provide information access to data to any citizen is obviously preferred, as common users are typically not familiar with technical query languages. Such a QA system should work over verified structured data storage, s.t., reasonable good quality can be ensured out of the box. Finally, the quality of such a QA system needs to be very high due to the official characteristics of the offered data.

To address these issues, we followed the Qanary methodology [1] for our implementation, as it enables traceability through a centralized tracing of the QA process and quality-control on the component level. The corresponding Qanary framework is intended to build structured data-based QA systems by reusing existing components. To store all information it uses the Resource Description Framework[1] (RDF) as internal knowledge representation. In this paper we describe our Coronabot Demonstrator (cf. Figure 1) that was developed in collaboration with the Federal Ministry of the Interior, Building and Community of Germany and the Informationstechnikzentrum Bund (short: ITZBund) which is the government-owned software development unit for the German authorities. It encapsulates data-driven functionality that is also integrated in the official COVID-19 chatbot of the German government[2] which previously only had supported FAQ-like functionality.

[1] cf. `https://www.w3.org/RDF/`
[2] `https://c19.bundesbots.de/`

2 Related Work

Dialogue systems are mainly divided into two classes: (1) task-oriented [6,5,7] and (2) general domain systems (like Amazon Alexa, Google Assistant, Yandex Alisa). In this work, we refer our system to the 1st class as its main goal is to consult people on COVID-19 questions. The architecture of data-driven dialogue systems is extended with one or more QA components, such as: query builder, document retriever, query executor, and passage retrieval (reader) component [3]. Typically, QA systems are separated into 2 main paradigms: (1) Knowledge Base QA (KBQA) and (2) Open Domain QA (OpenQA). The KBQA systems are designed to give precise answers to natural-language questions over structured data [2] while transforming questions into a corresponding query to a KB and then execute it in order to get an answer. End-to-end QA provides precise answers to natural language questions. However, such systems require a large set of training data as they are based on neural models and consequently are data-hungry. Modern examples of OpenQA systems are [9], and [8]. One of the major problems of all QA systems is that its components are being created from scratch all the time when a new system is developed [4]. To overcome the described problem, the Qanary framework was created [1]. The core concept of the framework is reducing recurring effort during the creation of QA systems. Instead of implementing QA components from scratch, they should be reused.

3 Approach and Implementation

The intended QA system is demanded to provide support on place-based questions (e.g., "How many cases were reported in August in *Dessau*?"), where even small places can be mentioned. Additionally, a time dimension is required, leading to the possibility of fetching data for a specific time span (e.g., "How many people died in *February*?"). Moreover, the system has to answer questions that combine place and time dimensions. It is worth to mention that the system has to work in two languages: German and English.

Following the Qanary methodology, typical tasks of the question processing are identified to model the components of the intended QA system. Each component will execute its dedicated task, i.e., creating new information about the given question and stores it in the Qanary triplestore which is representing the global memory of the RDF information computed while analyzing a question. For example, regarding the question "How many cases are reported in August in Dessau?" there are at least 3 RDF annotations that are created connected to the question resource established by the Qanary system: (1) there is a time-related statement from character index 29 to 37, and it is referring to August 2020, (2) at character index 42 to 47 a named entity is present that was linked to `https://www.wikidata.org/wiki/Q487070`, and (3) the whole question has the intent "case numbers". Hence, as (small) knowledge graph is created during the processing of each question. In the Qanary methodology, this RDF data is called *semantic annotations of the given question*.

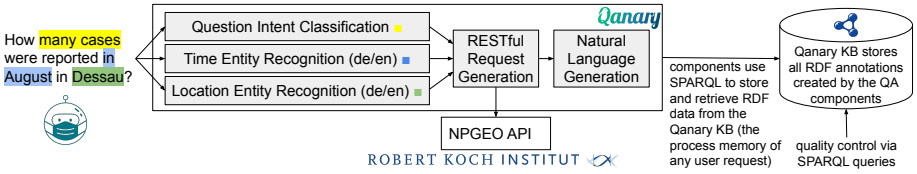

Fig. 2: High-level system overview (the actual system contains 11 components)

As the data is stored as RDF the annotations are modeled using the Qanary vocabulary which is a lightweight extension of the Web Annotation Data Model (WADM)[3]. While retrieving the stored annotations, the subsequently executed components of the QA process can retrieve any information that was previously stored in the Qanary triplestore for the given question. Finally, in our application, the data is fetched from an API[4] via a RESTful request to a Web service capable of providing the official data since the pandemic started.

According to this approach the following Qanary components were designed and implemented (see Figure 2): *Language Detection* as the component detecting whether a German or English question was asked, *Question Intent Classification* recognizes one 6 classes from the set: {infections, deaths} × {time, location, time and location}. *Location Entity Recognition* detects location entities and links the detected text span to the Wikidata ID[5], *Time Entity Recognition* recognizes time entities in the question and converts it to a normalized date representation, *RESTful Request Generation* generates and executes an NPGEO API request, *Natural Language Generation* provides a textual answer. The components are controlled by the pre-implemented Qanary pipeline (the reference implementation of the Qanary methodology) that is available as open source[6].

Hence, while a question is processed, an RDF graph is created reflecting all information that was computed within the QA process. To fulfill the quality demands an extensive test set of 18,000 questions were defined that is evaluated while reusing the RDF data stored in the Qanary triplestore, i.e., each test question is executed and thereafter several SPARQL queries are used to check if the expected semantic annotations were created in the virtual graph of the Qanary triplestore that is corresponding the current question. The evaluation results demonstrated that the system achieved an F1 score of 0.896 on English and 0.986 on German questions w.r.t. the predefined test set of location entities. Hence, by reusing background knowledge from the LOD cloud, an ad-hoc implementation of reasonable quality was possible.

[3] W3C Recommendation, 2017-02-23 (cf. `http://www.w3.org/TR/annotation-model`)
[4] NPGEO API of Robert Koch institute, the German federal government agency responsible for disease control and prevention: `https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/dd4580c810204019a7b8eb3e0b329dd6_0`
[5] The following Germany-specific entities are detected: federal states (16), counties (412), cities + communities (11806). The data was retrieved from the Wikidata KG.
[6] cf. `https://github.com/WDAqua/Qanary`

4 Conclusions

In this work, we showed a Coronabot Demonstrator based on a component-oriented architecture for creating dialogue systems over structured data driven by semantic technologies. The Coronabot extends the existing German government's chatbot functionality by providing a natural-language interface for data in two dimensions: date and location. The system was developed according to Qanary methodology, which demonstrated its flexibility for developing and evaluation processes. Since the QA system is RDF-based, it also offers the great advantage of processing traceability and thus end-to-end quality control. The demonstrator capturing the described functionality is available at `http://coronabot.ins.hs-anhalt.de/` the complete functionality is available as official Federal Coronabot at `https://c19.bundesbots.de/`. In the future, the implementation will be extended to cover additional knowledge domains of the Federal Government of Germany as well as additional generalized, reusable Qanary components that will be published as Open Source. Based on our outcomes, we believe that data-driven QA systems using semantic annotations have a clear perspective to become a development standard for QA processes.

References

1. Both, A., Diefenbach, D., Singh, K., Shekarpour, S., Cherix, D., Lange, C.: Qanary–a methodology for vocabulary-driven open question answering systems. In: European Semantic Web Conference. pp. 625–641. Springer (2016)
2. Cui, W., Xiao, Y., Wang, H., Song, Y., Hwang, S.w., Wang, W.: KBQA: learning question answering over QA corpora and knowledge bases. arXiv preprint arXiv:1903.02419 (2019)
3. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowledge and Information systems **55**(3), 529–569 (2018)
4. Singh, K., Radhakrishna, A.S., Both, A., Shekarpour, S., Lytra, I., Usbeck, R., et al.: Why reinvent the wheel: Let's build question answering systems together. In: Proceedings of the 2018 World Wide Web Conference. pp. 1247–1256 (2018)
5. Wei, Z., Liu, Q., Peng, B., Tou, H., Chen, T., Huang, X.J., Wong, K.F., Dai, X.: Task-oriented dialogue system for automatic diagnosis. In: Proceedings of the 56th Annual Meeting of the ACL (Vol. 2: Short Papers). pp. 201–207 (2018)
6. Wen, T.H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L.M., Su, P.H., Ultes, S., Young, S.: A network-based end-to-end trainable task-oriented dialogue system. arXiv preprint arXiv:1604.04562 (2016)
7. Yan, Z., Duan, N., Chen, P., Zhou, M., Zhou, J., Li, Z.: Building task-oriented dialogue systems for online shopping. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
8. Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., Lin, J.: End-to-end open-domain question answering with BERTserini. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations) (2019)
9. Yu, A.W., Dohan, D., Luong, M.T., Zhao, R., Chen, K., Norouzi, M., Le, Q.V.: QANet: Combining local convolution with global self-attention for reading comprehension. ArXiv **abs/1804.09541** (2018)