Trill: A reusable Front-End for QA systems

Dennis Diefenbach[1], Shanzay Amjad[2], Andreas Both[3], Kamal Singh[1], Pierre Maret[1]

[1] Laboratoire Hubert Curien, Saint Etienne, France,
dennis.diefenbach,kamal.singh,pierre.maret@univ-st-etienne.fr
[2] University of Ottawa, Ottawa, Canada,
samja088@uottawa.ca
[3] DATEV eG, Germany
contact@andreasboth.de

Abstract. The Semantic Web contains an enormous amount of information in the form of knowledge bases. To make this information available to end-users many question answering (QA) systems over knowledge bases were created in the last years. Their goal is to enable users to access large amounts of structured data in the Semantic Web by bridging the gap between natural language and formal query languages like SPARQL.
But automatically generating a SPARQL query from a user's question is not sufficient to bridge the gap between Semantic Web data and the end-users. The result of a SPARQL query consists of a list of URIs and/or literals, which is not a user-friendly presentation of the answer. Such a presentation includes the representation of the URI in the right language and additional information like images, maps, entity summaries and more.
We present Trill, the first reusable user-interface (UI) for QA systems over knowledge bases supporting text and audio input, able to present answers from DBpedia and Wikidata in 4 languages (English, French, German, and Italian). It is designed to be used together with Qanary, an infrastructure for composing QA pipelines. This front-end enables the QA community to show their results to end-users and enables the research community to explore new research directions like studying and designing user-interactions with QA systems.

Keywords: Question Answering Systems, Front-End, User Interaction, Answer Presentation

1 Introduction

Users' experience is an important factor for the success of a given application. Thus, the front-end of QA systems, which highly impacts the users' experience, is an important part of a QA system. On one side the research community has made a lot of efforts to increase the F-score over different benchmarks, i.e., the accuracy of the translation of a natural language question in a formal representation like a SPARQL query. On the other side not much attention has been paid to how the answer is presented and how users can interact with a QA system.
On a technical level one could say that a lot of attention has been paid to services in the back-end but little attention has been paid to the front-end. We hope to change this trend by presenting the first reusable front-end for QA systems over knowledge bases, i.e. a front-end that can be reused easily by any new QA system. It can currently be used for QA systems over DBpedia and Wikidata and supports 4 different languages.

2 Related work

In the last years a large number of QA systems were created by the research community. This is for example shown by the number of QA systems (more then 20 in the last 5 years) that were evaluated against the QALD benchmark [4]. Unfortunately only very few of them have a UI and even fewer are available as web services. Some exceptions that we are aware of are *SINA*[8], QAKiS[2] and *Platypus*[5]. Also industrial UI were created for well known systems like *Apple Siri*, *Microsoft Cortana*, *Google Search*, *Ok Google*, and *WolframAlpha*[6].
All the UIs mentioned above are tightly integrated with the corresponding QA systems. This is not the case for Trill. Trill can be used as a front-end for new QA systems. To achieve the modularity by making the front-end independent from the back-end of the QA system, we build Trill on top of a framework that provides a reusable architecture for QA systems. We are aware of four such architectures namely: QALL-ME [6], openQA [7], the Open KnowledgeBase and Question-Answering (OKBQA) challenge[7] and Qanary [9,1,4]. We choose to build Trill on top of the last mentioned framework, Qanary, since it allows the highest flexibility to integrate new QA components.

3 Description of Trill

In this section, we first describe the interaction of the front-end with the Qanary back-end. Then we describe which functionalities / presentational elements are implemented in the front-end. A screenshot showing most of the presentational elements is shown in Figure 1. A live demo can be found at `www.wdaqua.eu/qa`.
Trill builds on the APIs offered by Qanary. Qanary is a framework for composing QA pipelines. A QA system can be seen as a pipeline, i.e., as a sequence of services that are called one after the other and finally are able (or not) to compute the answer for a given question. Qanary has APIs that allow to combine components integrated in Qanary starting from a text or an audio question. The human accessible versions are exposed for example under:

`www.wdaqua.eu/qanary/startquestionansweringwithtextquestion`,
`www.wdaqua.eu/qanary/startquestionansweringwithaudioquestion`

A call to these APIs require as parameters both the question and a list of Qanary components. A call will activate the mentioned components which will be executed as a pipeline in the defined order. Trill accesses the Qanary back-end using these APIs. In particular the demo running under `www.wdaqua.eu/qa` calls a Qanary component called WDAqua-core0 [5] created in the frame of the WDAqua project[8].
Now we give a list of all the functionalities and presentational elements that are available in Trill. The numbers in the item list correspond to the numbers in Figure 1.

[4] `http://qald.sebastianwalter.org`
[5] `http://sina.aksw.org`, `http://live.ailao.eu`, `http://qakis.org/qakis2/`, `https://askplatyp.us/?`
[6] `http://www.apple.com/ios/siri`, `http://www.wolframalpha.com`, `http://windows.microsoft.com/en-us/windows-10/getstarted-what-is-cortana`, `www.google.com` `https://support.google.com/websearch/answer/2940021?hl=en`
[7] `http://www.okbqa.org/`
[8] `http://www.wdaqua.eu`

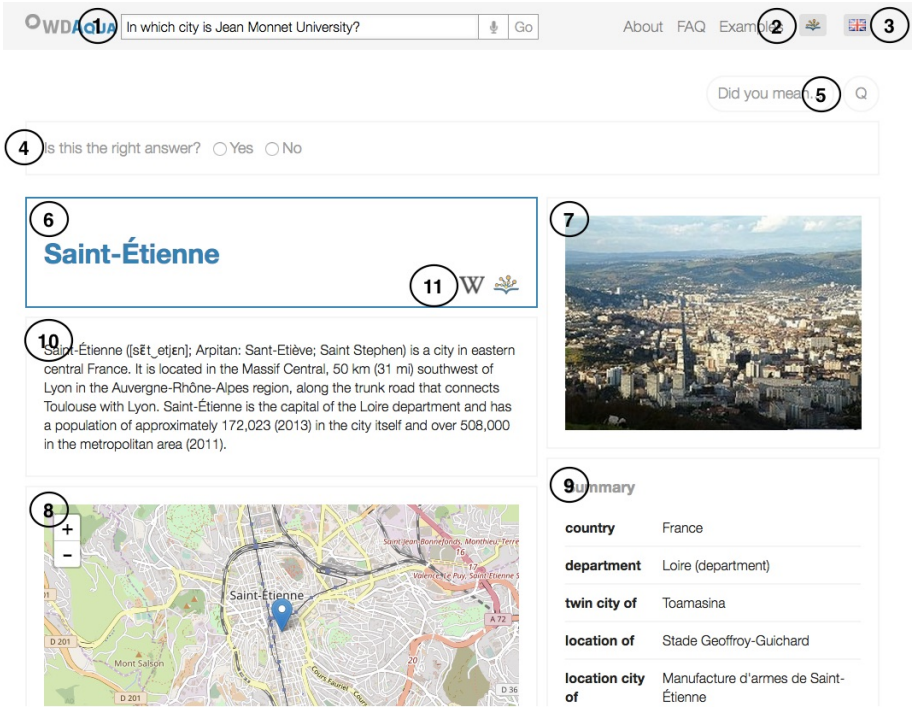

Fig. 1. Screenshot of Trill for the question "In which city is Jean Monnet University?". The numbers refer to the item list in Section 3

1. **Text and audio input**: The front-end contains a search-bar component. This allows the user to input a text question using a text bar or a vocal question using a microphone[9].
2. **Language selection**: The user has the possibility to change between 4 languages (English, French, German, and Italian) by selecting the flag of the corresponding country. The whole webpage will re-render using labels of the selected language and also the answers to a question will be adapted to the corresponding language.
3. **Knowledge-base selection**: The user has the option to choose an available knowledge base to query. Actually this is limited to DBpedia and Wikidata. Note that the answer presentation is coupled to the knowledge base since different properties are used to express the same information, i.e., the image can be retrieved in DBpedia using the property `dbo:thumbnail` while in Wikidata it is the property `wdt:P18`.
4. **Feedback functionality**: We implemented an easy-to-use feedback functionality where a user can choose if the answer is correct or not. This information is logged in the back-end and can be used to create a training set.
   Answer presentation We implemented several components to display information about the answer that can be useful for a user. Here is a list.

[9] This feature is supported only for Chrome and Firefox due to dependencies on the MediaStream Recording API.

5. **Generated query**: For expert users, we give the possibility to see the generated SPARQL query for the question.
6. **Label** In the case the answer is a URI, we display its label in the corresponding language, i.e., `wd:Q90` will be displayed as "Paris" in English and "Parigi" in Italian. If the answer is a literal, we display it directly.
7. **Image** One component retrieves the image (if it exists) of the answer and displays it.
8. **Map** If available the geo-coordinates are used to display the position of the answer on a map. The map is showed using Leaflet[10] Javascript Library which retrieves maps from OpenStreetMap[11].
9. **Top-k Properties** A summary of the entity is visualized using an external service called LinkSUM[10] provided by Andreas Thalhammer.
10. **Text description from Wikipedia abstract**: Every entity in DBpedia is linked to the corresponding Wikipedia page and for many Wikidata entities this is also the case. Following these links we use the Wikipedia API to retrieve the introduction of the article.
11. **External Links** If available, we present some external links to the user. This include links to DBpedia, Wikidata and Wikipedia.
12. **Ranking of list answers** In the case of a list of answers, we rank them in the case of DBpedia. For example, if a user asks about "italian lakes" the result will be a long list of answers. Without ranking, the first answer will probably be a small unknown lake in Italy. For this reason we reorder the results using the page rank scores presented in [11] that are available online[12]. Hence, if a user asks for "italian lakes", he will get at a first position in the result list the entity "Lake Como" followed by "Lake Maggiore".

The code can be found under `https://github.com/WDAqua/Trill`. Table 1 compares the features of other open source front-ends with Trill.

	Audio input	Language sel.	KB sel.	Feedback func	Generated query	Entity label	Image	Map	Top-k properties	Text description	External links	Language support	KB support
SINA						x	x	x		x	x	en	dbpedia
QAKiS			x			x	x				x	en	dbpedia
Platypus	x					x	x	x		x	x	en	wikidata
Trill	x	x	x	x	x	x	x	x	x	x	x	en, de, fr, it	dbpedia, wikidata

Table 1. Table comparing the features of Trill with other open source UIs.

[10] http://leafletjs.com
[11] http://openstreetmap.org/
[12] http://people.aifb.kit.edu/ath/#DBpedia_PageRank

4 Conclusion

We have presented Trill, a reusable user-interface for QA systems over knowledge bases that can be used on top of Qanary. While many efforts were made to develop better QA technology in the back-end, little work was done in the front-end.
We hope that Trill can change this trend. Trill can be used for any future QA pipeline integrated into Qanary as an off-the-shelf front-end. Moreover, it can be used to deeply study the interactions of the end-users with QA systems. This includes: 1. collect end-user queries to create easily large and realistic benchmarks, 2. studies to analyze which questions users ask, 3. study how much context information should be presented to a user together with the answer, 4. create interfaces, like the disambiguation interfaces in [3], to allow users to interact with QA systems. These examples shows how advances on the front-end can also be beneficial for classical QA research in the back-end.

Acknowledgments Parts of this work received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 642795, project: Answering Questions using Web Data (WDAqua).

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: ESWC 2016 (2016)
2. Cabrio, E., Cojan, J., Aprosio, A.P., Magnini, B., Lavelli, A., Gandon, F.: Qakis: an open domain qa system based on relational patterns. In: Proceedings of the 2012th International Conference on Posters & Demonstrations Track-Volume 914 (2012)
3. Diefenbach, D., Hormozi, N., Amjad, S., Both, A.: Introducing feedback in qanary: How users can interact with qa systems. In: ESWC P&D (2017)
4. 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)
5. Diefenbach, D., Singh, K., Maret, P.: Wdaqua-core0: A question answering component for the research community. In: ESWC, 7th Open Challenge on Question Answering over Linked Data (QALD-7) (2017)
6. Ferrández, Ó., Spurk, C., Kouylekov, M., Dornescu, I., Ferrández, S., Negri, M., Izquierdo, R., Tomás, D., Orasan, C., Neumann, G., Magnini, B., González, J.: The QALL-ME framework: A specifiable-domain multilingual Question Answering architecture. J. Web Sem.
7. Marx, E., Usbeck, R., Ngonga Ngomo, A., Höffner, K., Lehmann, J., Auer, S.: Towards an open question answering architecture. In: SEMANTiCS (2014)
8. 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)
9. Singh, K., Both, A., Diefenbach, D., Shekarpour, S.: Towards a message-driven vocabulary for promoting the interoperability of question answering systems. In: ICSC 2016 (2016)
10. Thalhammer, A., Lasierra, N., Rettinger, A.: Linksum: using link analysis to summarize entity data. In: International Conference on Web Engineering. Springer (2016)
11. Thalhammer, A., Rettinger, A.: Pagerank on wikipedia: towards general importance scores for entities. In: International Semantic Web Conference. Springer (2016)