Toward a Voice Querying on Smart Speaker via SPARQL

Takuya Adachi[1] and Naoki Fukuta[2]

[1] Department of Informatics, Graduate School of Integrated Science and Technology, Shizuoka University
[2] College of Informatics, Academic Institute, Shizuoka University
{adachi.takuya.17@, fukuta@inf.}shizuoka.ac.jp

Abstract. In this paper, we present a mechanism and its prototype design to realize speak to query interface for querying SPARQL queries via smart speakers. Our prototype system allows us to retrieve results from an endpoint as a voice command. Additionally, our approach tries to create the answers for the given SPARQL queries by voice commands via smart speakers.

Keywords: SPARQL, smart speaker, ontology mapping

1 System Overview

We often ask something to a smart speaker device using a natural language query. However, it is not an easy way to clearly express some complex conditions of queries using such a natural language query. SPARQL is a query language to communicate to various SPARQL endpoints by a software. However, there is nothing to limit such communications to the way to communicate among software systems, and some people who are familiar with SPARQL actually speak SPARQL for communication. In this paper, we present a mechanism to help such people better utilize current smart speaker devices.

Figure 1 shows an overview of our prototype system design. A Users speaks a SPARQL query to a smart speaker, then the voice activity detector analyzes the user's voice and prepares the SPARQL query. The query executor retrieves results from SPARQL endpoints by using a SPARQL query to fit to the spoken content. The response generator interprets results and generates a response to respond the result in a speaker-friendly form. After that, the voice synthesizer outputs the actual auditory data generated by the speech synthesis engine to a smart speaker. In this way, the smart speaker tells the result to the users.

Figure 2 shows a system structure of our prototype system design. Smart speakers could be some of commercially sold products such as Google Home, Amazon Echo (dot), Xperia Hello!. Smart speakers have some functions to recognize the user's voice and to play an audio data according to the commands from the user. Smart speakers often give users to develop applications themselves. The voice assistant can be connected to commercial products such as

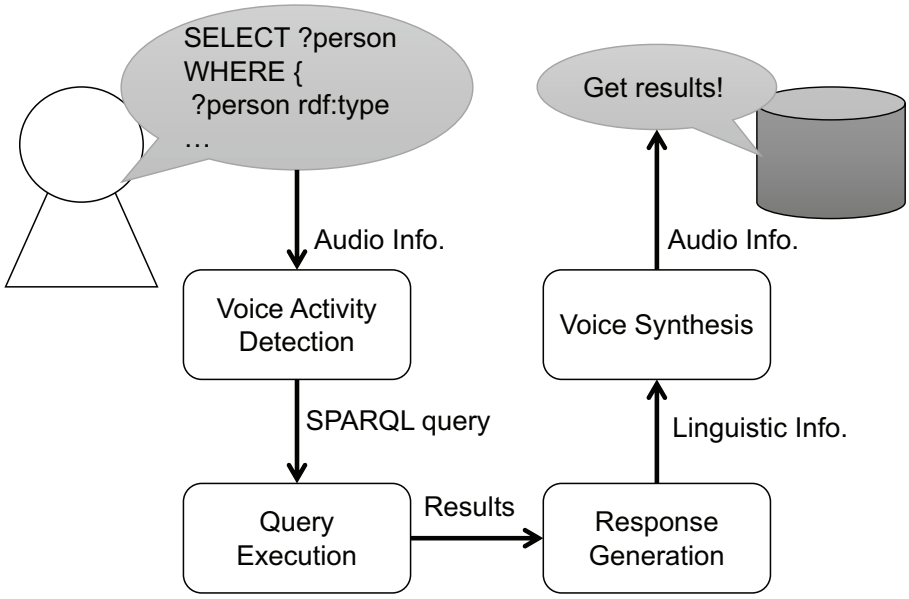

Fig. 1. An overview of our prototype system

Google Assistant[3][3]. The action executor selects an action event when that gets an analysis result from the voice assistant and sends the results from external web apis to the voice assistant. The action executor can be connected to some process control mechanisms such as Dialogflow[4]. Our Web API has features about parsing and reconstructing queries from voice commands, managing query executions and the response generations.

Figure 3 shows an example smart speaker device for our prototype system. On our prototype implementation, we utilize Amazon Echo dot with Amazon Alexa[5].

2 Response Generation Mechanism

SPARQL has four query forms: *SELECT*, *CONSTRUCT*, *ASK* and *DESCRIBE*. These forms will return the different types of results. To generate an appropriate response that returns results of the SPARQL query to users, our mechanism selects the response generation mechanism for each SPARQL query form.

On responding to SELECT form, the response generator generates two information. one is the number of results, another is each result. If the result is large[6], then a response uses a few results. For example, the response generator

[3] https://assistant.google.com/
[4] https://dialogflow.com/
[5] https://developer.amazon.com/alexa
[6] The large means the size is exceeding a predefined threshold.

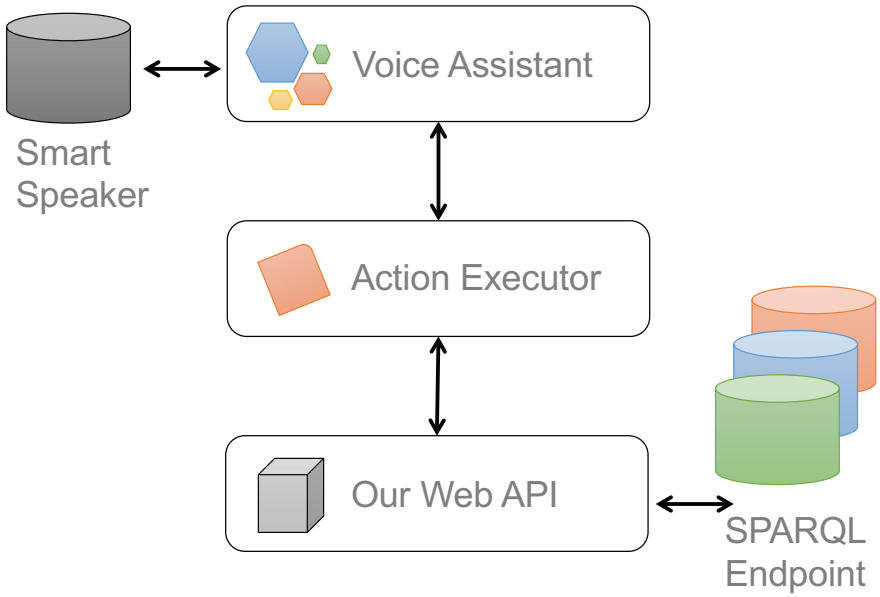

Fig. 2. A system structure of our prototype system

gets 10 results, then the response should be like "I gets 10 results, I read the first 5 results". On responding to CONSTRUCT form, the response contains what triples are constructed since the user might seek them. On responding to ASK form, the response generator creates the answer, that contains not only "Yes" or "No" but also the number of results. On responding to DESCRIBE form, the response is what results are obtained in detail, that contain the number of entities connected properties.

Results sometime contain URIs. If a smart speaker speaks a URI, it is difficult for users to understand what URI means. Our mechanism obtains labels and comments of URIs to understand URIs and speak them[7].

3 Empty Answer Problem

When a user speaks SPARQL queries, then the user needs precisely understanding the structure and schema of the dataset in the endpoint. Moreover, it is hard for the user to fully speak URIs in SPARQL queries. In the case that an inappropriate query returns an empty set, that is called empty-answer problem.

There are some effective approaches to be solved this problem (for example, graph embedding approach[4]). Although these approaches are helpful in some context, in this case that the user speak SPARQL queries, these approaches

[7] Applying some ontology mapping techniques to obtain better understandable labels is future work.

Fig. 3. An example smart speaker device

might not be helpful enough. On our prototype system, in the case the query returns an empty set, then we try to execute enhanced SPARQL queries using SPARQLoid mechanism[2] and on-the-fly ontology matching mechanism[1] to retrieve user seeking results.

4 Conclusion

In this paper, we presented a mechanism and its prototype design to realize speak-to-query interface for directly querying SPARQL queries via smart speakers. Our prototype system allows us to retrieve results from an endpoint as a voice command. Additionally, our approach tries to create the answers for the given SPARQL queries by voice commands via smart speakers.

In our future work, we consider a way to address the issue about how to represent graphical structures in auditory from in combination to some display devices for representing more complex results obtained from complex queries. To fully integrate our mechanism to many smart speakers is also future work.

References

1. Adachi, T., Fukuta, N.: A Mapping-enhanced Linked Data Inspection and Querying Support System using Dynamic Ontology Matching. In: Proc. of 2nd International Workshop on Platforms and Applications for Social problem Solving and Collective Reasoning (PASSCR2017). pp. 1191–1194 (2017)
2. Fujino, T., Fukuta, N.: Utilizing Weighted Ontology Mappings on Federated SPARQL Querying. In: Proc. of the 3rd Joint International Semantic Technology Conference (JIST2013) (2013)
3. Li, B., Sainath, T., Narayanan, A., Caroselli, J., Bacchiani, M., Misra, A., Shafran, I., Sak, H., Pundak, G., Chin, K., Sim, K.C., Weiss, R.J., Wilson, K., Variani, E., Kim, C., Siohan, O., Weintraub, M., McDermott, E., Rose, R., Shannon, M.: Acoustic modeling for google home. In: INTERSPEECH 2017 (2017)
4. Wang, M., Wang, R., Liu, J., Chen, Y., Zhang, L., Qi, G.: Towards Empty Answers in SPARQL: Approximating Querying with RDF Embedding. In: Proc. of the 17th International Semantic Web Conference (ISWC2018). pp. 513–529 (2018)