Dedicated Workflow Management for OKBQA Framework

Jiseong Kim, GyuHyeon Choi, Key-Sun Choi
Machine Reading Laboratory
Semantic Web Research Center
Department of Computer Science
KAIST, Daejeon, Korea
{jiseong, wiany11, kschoi}@kaist.ac.kr

Abstract

Nowadays, a question answering (QA) system is used in various areas such a quiz show, personal assistant, home device, and so on. The OKBQA framework supports developing a QA system in an intuitive and collaborative ways. To support collaborative development, the framework should be equipped with some functions, e.g., flexible system configuration, debugging supports, intuitive user interface, and so on while considering different developing groups of different domains. This paper presents OKBQA controller, a dedicated workflow manager for OKBQA framework, to boost collaborative development of a QA system.

1 Introduction

Recently, a QA system have been on the rise being applied to diverse domains, e.g., quiz show (IBM Watson), personal assistant (Apple Siri, Microsoft Cortana), home device (Amazon Echo), and so on.

To make a QA system, the OKBQA framework focuses on constructing an OKBQA pipeline-based QA system. The OKBQA pipeline is based on the state-of-the-art researches such as template generation (Unger et al., 2012), disambiguation (Usbeck et al., 2014), query generation (Kim and Cohen, 2014), and so on, which is depicted in Figure 1.

The main goal of the OKBQA framework is to support **collaborative development** of an OKBQA pipeline-based QA system, To support the collaborative development, the framework should be equipped with key functions:

- **Pipeline construction based on OKBQA specification** As modules of the OKBQA pipeline are developed by different groups of different domains, I/O specification is crucial to integrate modules developed independently into an integrated whole system. The OKBQA specification specifies that an I/O format of OKBQA module should be a JSON format and their interface should be implemented as a REST API. That is, the (OKBQA) framework should be capable of linking modules of JSON-formatted I/O with a RESTful service. By compliance with the OKBQA specification, modules developed by a different groups can be integrated into one QA system.

- **Flexible pipeline configuration** By open collaboration, an QA system can be constructed by modules developed by different developers. To support a developer who wants to construct his QA system by reusing some modules developed by other developers, the framework should be equipped with the function of configuring which modules will compose his QA system.

- **Debugging supports** As different users can develop a module of a QA system independently, some modules can cause a crash of an entire system by diverse errors. To support developers chasing a cause of errors, the framework should be capable of showing exceptional information about which module is crashed, the input causing the crash, and the reason why the module is crashed.

- **Intuitive user interface** To support developers of diverse domains, the framework should provide an intuitive and common user interface that can lower the entry barrier of QA system development.

Proceedings of the Open Knowledge Base and Question Answering (OKBQA) Workshop,
pages 97–101, Osaka, Japan, December 11 2016.

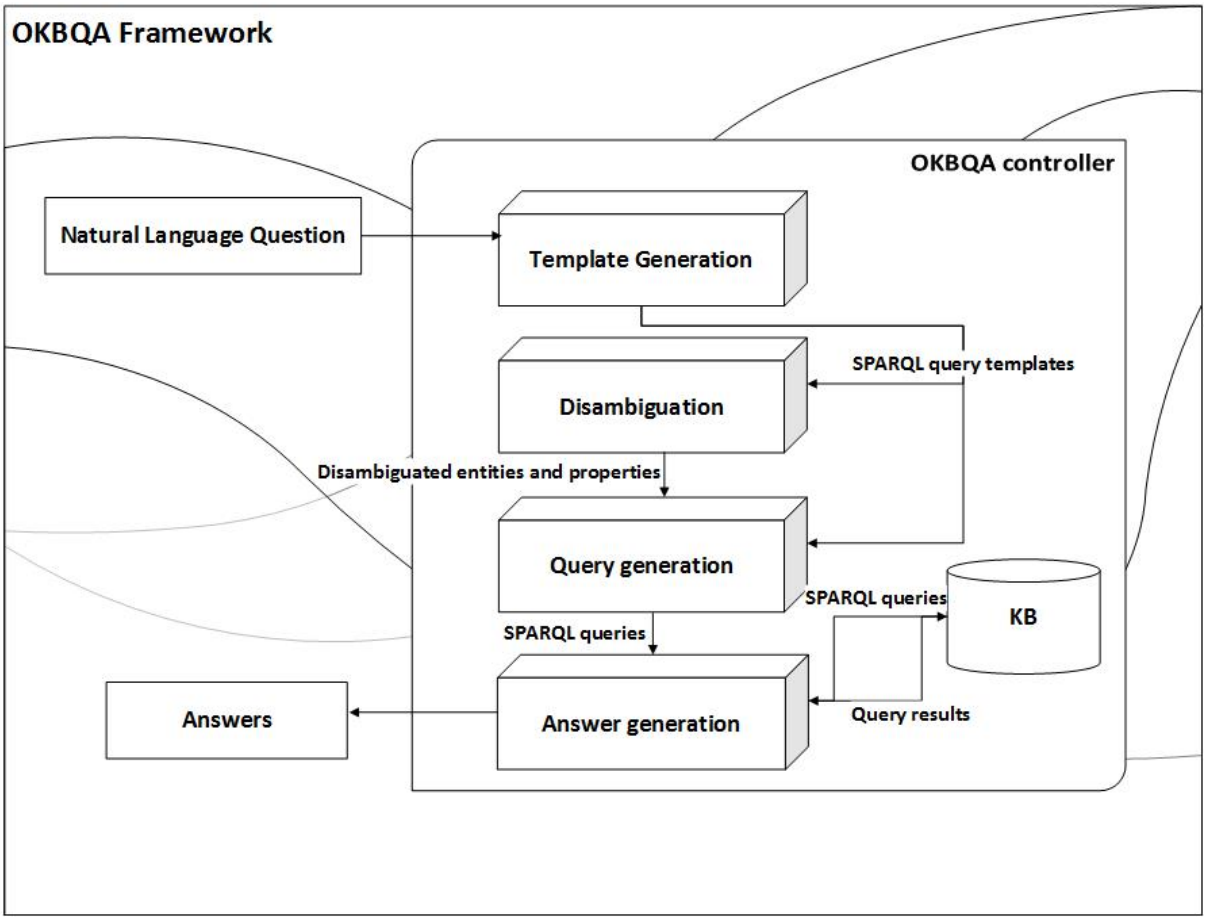

Figure 1: Workflow of an OKBQA pipeline

The above-mentioned functions are traditionally dealt by a workflow manager, which is a kind of a module for linking other modules to construct an integrated system. In this paper, we present a dedicated workflow manager for the OKBQA framework, so-called **OKBQA controller**, to boost the collaborative development of an OKBQA pipeline-based QA system.

2 OKBQA Controller

The OKBQA controller is a dedicated workflow manager for constructing a OKBQA pipeline by linking OKBQA modules as shown in Figure 1. The controller makes a pipeline work by transferring I/O of each module sequentially. The controller realizes and provides the key functions described in Section 1, which is detailed in the following sections.

2.1 Pipeline construction based on OKBQA specification

The controller makes a pipeline work consistently with the OKBQA specification by linking RESTful modules of JSON-formatted I/O. The I/O of the controller, depicted in Figure 2, are also compliant with the OKBQA specification as other OKBQA modules; The controller's I/O have a JSON format and interface is implemented in a RESTful service. By compliance with the OKBQA specification, modules can be developed in a consistent ways w.r.t. their I/O and interface implementation, so the reusability of modules can be significantly enhanced.

2.2 Flexible pipeline configuration

To support constructing a pipeline with various structure and composition of modules and reusing modules developed by other developers, the controller supports configuring addresses and executing sequence

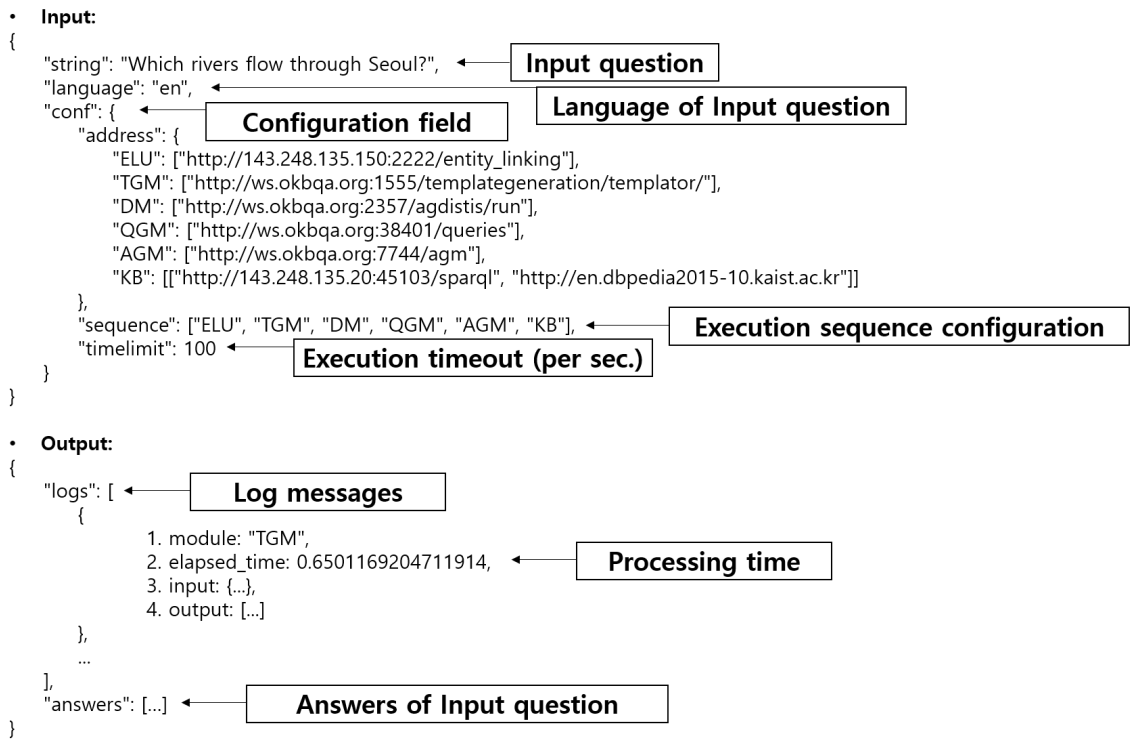

Figure 2: I/O of an OKBQA controller

of modules by the controller's input fields "address" and "sequence" as shown in Figure 2. By configuring the number and executing sequence of modules, developers can construct their own pipeline different from the original OKBQA pipeline to apply new idea and improve their own QA system further; For example, one idea is that if disambiguated results are provided to a template generation process as an input, there is a possibility that results of a template generation module could be improved.

2.3 Debugging supports

To support efficient debugging for collaborative development, the controller provides a fault alarming function through a log message that is the field "log" in controller's output as shown in Figure 2. The log message provides the information such as input, output, and processing time of each module, name of module throwing exception, a cause of exception, and so on; these information can be useful for chasing a cause of errors that are caused by not only our module, but also the others'. When a module throws an exception, the controller will stop executing a pipeline and return an exceptional message on log; e.g. Figure 3 shows an example of the message. By the message, developers can easily notice which module is problematic and what to do for fixing it. It is an essential function to easily chase and fix errors caused by modules developed by different developers.

2.4 Web-based user interface

The controller provides a Web-based user interface[1] to developers as shown in Figure 4. Through the graphical supports by the interface, developers can set a system configuration, integrate their modules with other developers' modules to construct an integrated QA system, and test constructed QA system by asking the pre-defined natural language questions in an easy and intuitive way.

2.5 Conclusion

We have presented a dedicated workflow manager for the OKBQA framework, so-called OKBQA controller. We showed that the OKBQA controller has an essential functions to develop a QA system in a

[1] `http://ws.okbqa.org/web_interface`

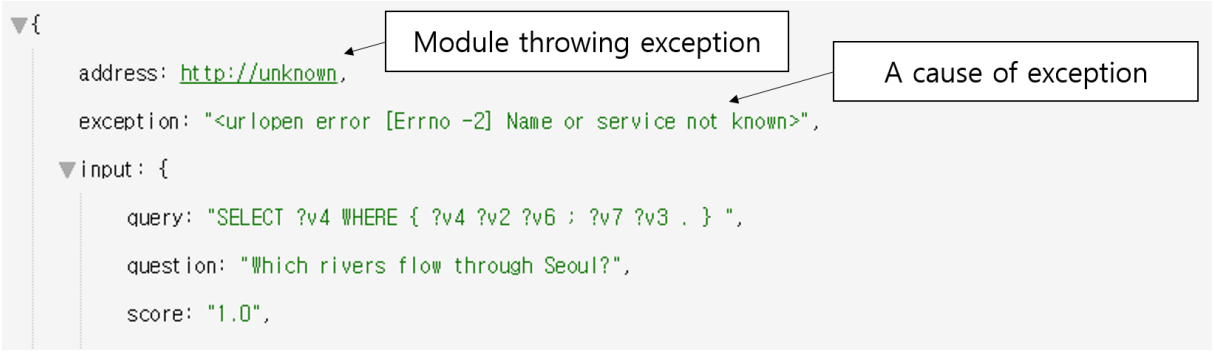

Figure 3: An example of an exceptional message

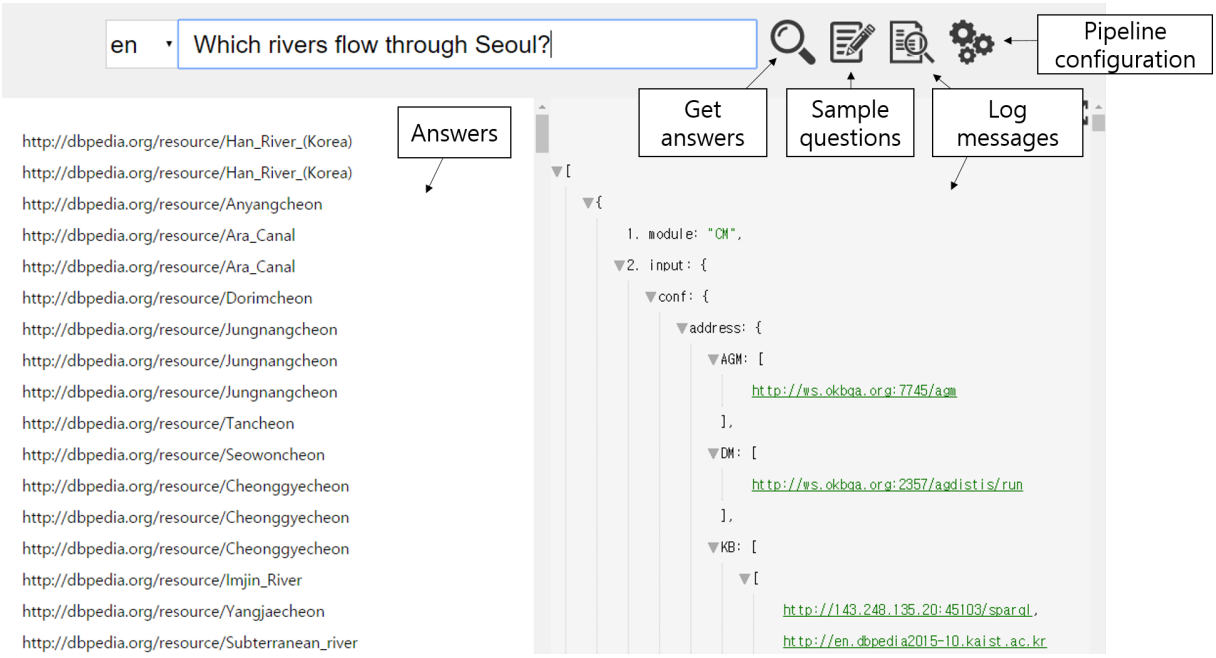

Figure 4: A Web-based user interface for an OKBQA controller

collaborative way. However, there are some points to be improved and further developed. We will keep searching needs of developers and mirroring their needs to our successive versions of the controller.

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. R0101-16-0054, WiseKB: Big data based self-evolving knowledge base and reasoning platform). And also this work was supported by the Bio & Medical Technology Development Program of the NRF funded by the Korean government, MSIP(2015M3A9A7029735).

References

Unger, Christina and Bühmann, Lorenz and Lehmann, Jens and Ngonga Ngomo, Axel-Cyrille and Gerber, Daniel and Cimiano, Philipp. 2012. *Template-based question answering over RDF data. Proceedings of the 21st international conference on World Wide Web* (pp. 639–648). ACM.

Usbeck, Ricardo and Ngomo, Axel-Cyrille Ngonga and Röder, Michael and Gerber, Daniel and Coelho, Sandro Athaide and Auer, Sören and Both, Andreas. 2014. *AGDISTIS-graph-based disambiguation of named entities using linked data. International Semantic Web Conference* (pp.457-471). Springer.

Kim, J. D. and Cohen, K.. 2014. *Triple pattern variation operations for flexible graph search*. *Workshop on Natural Language Interfaces for Web of Data*.