DialBB: A Dialogue System Development Framework as an Educational Material

Mikio Nakano[1,2] and Kazunori Komatani[2]

[1]C4A Research Institute, Inc., 1-13-12 Umegaoka, Setagaya, Tokyo, Japan
[2]SANKEN, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka, Japan
mikio.nakano@c4a.jp, komatani@sanken.osaka-u.ac.jp

Abstract

We demonstrate DialBB, a dialogue system development framework, which we have been building as an educational material for dialogue system technology. Building a dialogue system requires the adoption of an appropriate architecture depending on the application and the integration of various technologies. However, this is not easy for those who have just started learning dialogue system technology. Therefore, there is a demand for educational materials that integrate various technologies to build dialogue systems, because traditional dialogue system development frameworks were not designed for educational purposes. DialBB enables the development of dialogue systems by combining modules called building blocks. After understanding sample applications, learners can easily build simple systems using built-in blocks and can build advanced systems using their own developed blocks.

1 Introduction

To build a dialogue system, it is generally necessary to adopt an appropriate architecture according to the application and integrate various technologies. While the advancements in large language models have led some to believe that dialogue systems can be developed solely with these models and that the developers do not need to know about architecture and elemental technologies, there are issues such as hallucinations, so it is not always practical to build dialogue systems with only large language models, depending on the application.

Over the years, research into dialogue systems has evolved, accumulating knowledge on what kind of dialogue systems should be built with what technologies and what architectures. However, it is not easy for people who are learning dialogue system technology to acquire this knowledge. An educational material that allows people to learn dialogue system technology while building various dialogue systems would be helpful.

In learning about dialogue system technology, it is important to understand various elemental technologies such as language understanding and dialogue management, as well as an architecture based on appropriate modularization, extensibility which facilitates improving systems, and domain portability. It is also crucial to understand the importance of robustness in intention understanding and interaction design through running actual dialogue systems.

As an educational material that is useful for such learning, a dialogue system development framework with the following features is beneficial: (1) including various elemental technologies of dialogue systems, (2) appropriately modularized, (3) highly extensible, (4) including sample applications that help learners' understanding of dialogue system technology, and (5) making it possible to develop simple applications without extensive skills or knowledge in system development, enabling the acquisition of various technologies while improving the system. In addition, it is desired that its source code is available.

There are several dialogue system development tools whose source codes are available. PyDial (Ultes et al., 2017), OpenDial (Lison and Kennington, 2016), ConvLab-3 (Zhu et al., 2023), and ADVISER (Ortega et al., 2019) focus on statistical dialogue models for task-oriented dialogue systems, while we think educational materials should support state-transition network-based dialogue management which is often used for building practical dialogue systems. Although Rasa Open Source (Bocklisch et al., 2017) is highly extensible, it does not support state-transition network-based dialogue management by default. Botpress[1] supports state-transition network-based dialogue management, but replacing its internal modules with custom-made ones is not easy. MMDAgent (Lee et al.,

[1]https://botpress.com

Proceedings of the 25th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 664–668
September 18–20, 2024.

2013) also supports state-transition network-based dialogue management, but it is not easy to extend it.

We have been building a dialogue system development framework called **DialBB** (*Dial*ogue system development framework with *B*uilding *B*locks)[2] intended for use as an educational material in dialogue system development. DialBB is written in Python, and supports the development of English and Japanese applications.

2 Overview of DialBB

Here, we give an overview of DialBB. For more details, please refer to its document.[3]

2.1 Architecture

DialBB allows the development of dialogue systems by combining modules called Building Blocks (hereafter referred to as "blocks"). Figure 1 shows the architecture of DialBB applications.

The main module of DialBB works as follows. First, it receives input containing user utterances in JSON format through a method call of the class API or via a Web API. This input is then stored in the blackboard.[4] Next, it calls each block in the order specified in the configuration file, using parts of the blackboard as input for these blocks. The output from the blocks is used to update the blackboard. By sequentially driving each block in this manner, the system generates and returns a response. Additionally, the input and output of the main module can include not only utterance strings but also additional information, so that it is possible to handle multimodal information such as speech recognition confidences, user emotion estimation results, and gesture commands.

Which blocks each application uses is specified by describing the block classes in the application's configuration file (a YAML file). The configuration file also specifies what type of data each block receives and sends. Furthermore, the values of parameters used within the blocks and the knowledge description files used by the blocks can also be specified in the configuration file.

[2]DialBB is publicly available for non-commercial use at https://github.com/c4a-ri/dialbb. This paper is based on its ver. 0.8.

[3]https://c4a-ri.github.io/dialbb/document-en/build/html/

[4]We call it a 'blackboard' in analogy to the blackboard model (Erman et al., 1980), but unlike the blackboard model, each block is called in the order written in the configuration file.

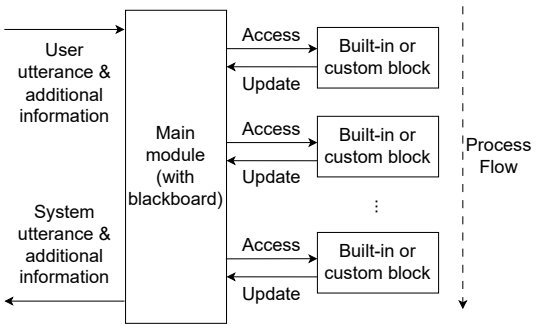

Figure 1: Architecture of DialBB applications.

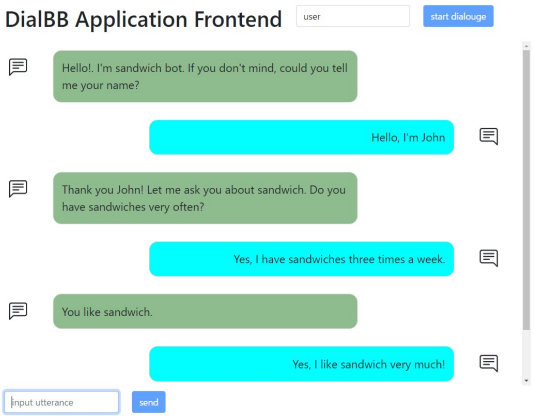

Figure 2: A snapshot of the frontend.

DialBB includes a frontend for engaging in dialogues via a Web API (Figure 2).

2.2 Built-in Blocks

To make it easier for learners to build conversational systems, DialBB has built-in building blocks listed in Table 1. For simplicity, only those for English applications are listed. Below we explain knowledge to be written by developers for use in some built-in blocks.

Language Understanding Knowledge Snips Understander Block and ChatGPT Understander Block use language understanding knowledge which consists of a collection of utterance examples that are annotated with intents and slots like the following.

Intent	Example utterance
tell-favorite-sandwich	I love (chicken salad sandwiches) [favorite-sandwich]
acknowledge	Definitely

Here, "[favorite-sandwich]" indicates a slot name, and "(chicken salad sandwiches)" indicates a slot

Block	Input	Output	Task
Simple Canonicalizer	user utterance string	canonicalized user utterance string	Canonicalizes the input string (convert uppercase to lowercase, etc.).
Whitespace Tokenizer	canonicalized user utterance string	token list	Performs tokenization based on white spaces.
Snips Understander[5]	token list	intent and slots	Performs language understanding using Snips NLU (Coucke et al., 2018) to obtain the intent and slots.
ChatGPT Understander	user utterance string	intent and slots	Performs language understanding using the JSON mode of OpenAI's ChatGPT.[6]Creates few-shot examples to embed in prompts from language understanding knowledge.
spaCy-Based NER	user utterance string	named entities	Performs named entity recognition using spaCy.[7]
STN Manager	user utterance string, intent, slots, and named entities (all are optional)	system utterance string	Manages dialogues using a state-transition network.
ChatGPT Dialogue	user utterance string	system utterance string	Generates a system utterance using ChatGPT based on a prompt including system persona, situation, and dialogue history.

Table 1: List of built-in blocks. Only important inputs and outputs are shown.

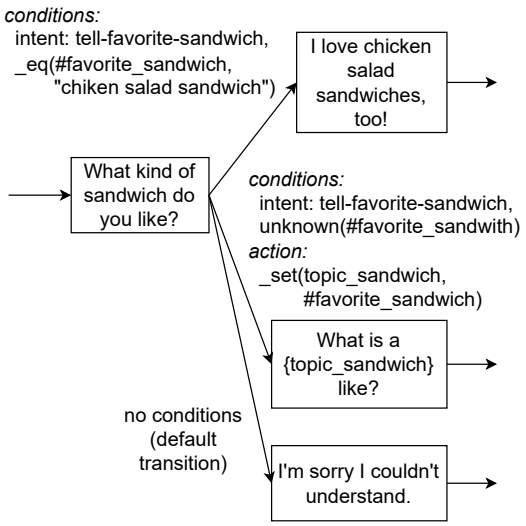

Figure 3: A part of an example state-transition network. Transitions above are given priority.

value. In addition, the knowledge used for language understanding includes a list of synonyms.

State-Transition Network STN Manager block uses a state-transition network (STN, also called a *scenario*). State-transition network-based dialogue management is often used in practical dialogue systems. Figure 3 illustrates a part of the state-transition network. Each state is assigned a system utterance that is generated in that state. The state transitions to another state according to the input user utterance. Transitions can be accompanied by conditions for the transition and actions to be performed during the transition. Conditions are based on the intent of the user utterance and function calls. Actions are function calls. Functions used in conditions and actions are referred to as *scenario functions*. Within the definition of scenario functions, it is possible to use slots extracted in language understanding and named entity recognition results (for example, `#favorite_sandwich` in Figure 3 is a slot value). Scenario functions can also access *contextual information*, which consists of data that the system remembers as the dialogue progresses, such as user requests and preferences (`topic_sandwich` in Figure 3 is an example of this). Contextual information can be used in system utterances, as in "`What is a {topic_sandwich} like?`" in Figure 3. It is also possible to access separately operating databases or external APIs in scenario functions.

Additionally, STN Manager block includes built-in scenario functions, which can reduce the effort of defining functions. For instance, there is a built-in function that compares if strings are identical (`_eq` in Figure 3), and a built-in function that sets a value to a variable (`_set` in Figure 3). Furthermore, there are built-in functions that use ChatGPT. One

[5]Snips Understander Block will be deprecated in DialBB ver. 0.9 because Snips does not work with Python 3.9+. Instead DialBB ver. 0.9 will have a built-in block for language understanding that utilizes logistic regression and conditional random fields.

[6]https://openai.com/index/chatgpt

[7]https://spacy.io/

is for determining if conditions written in natural language (e.g., "Is the user bored with the conversation?") are satisfied, and the other is for generating utterance strings based on instructions written in natural language (e.g., "Generate a response to the user's utterance in less than 30 words"). These functions call ChatGPT by incorporating into the prompt the dialogue history and the situation and persona settings specified in the configuration file.

Language understanding knowledge and state-transition networks can be described using spreadsheets.

STN Manager block has additional functionalities for handling speech recognition results. A DialBB application can receive speech recognition confidence together with the speech recognition result of the user utterance. STN Manager block can make an utterance to ask for repetition or request confirmation depending on the configuration. It is also possible to process barge-in utterances differently from ordinary utterances. In addition, reacting to a long silence after a system utterance is possible.

2.3 Sample Applications

DialBB has several sample English and Japanese applications that use only these built-in building blocks. Below are English applications.

Snips+STN Application uses Simple Canonicalizer, Whitespace Tokenizer, Snips Understander, and STN Manager blocks and it can engage in a simple dialogue about sandwiches.

Lab Application uses Simple Canonicalizer, ChatGPT Understander, spaCy NER, and STN Manager Blocks and it can also engage in a simple dialogue about sandwiches, but it demonstrates various advanced features of the built-in blocks.

ChatGPT Application uses only ChatGPT Dialogue block. It can engage in a dialogue using ChatGPT based on a prompt template that describes the dialogue situation and system persona.

To serve as a reference for learners, these sample applications, the built-in blocks, and the main module of DialBB are written in code that is as readable as possible.

2.4 Custom Blocks

Developers can create and use their own custom blocks. A block's class can be created by inheriting from an abstract class `AbstractBlock` and implementing the necessary methods. The created class can then be specified in the configuration file for use. This enables using different language understanding and dialogue management than those of built-in blocks.

3 Learning Dialogue System Technology Using DialBB

Using DialBB, learners can learn about dialogue system technology through the following steps. First, by understanding the sample applications, they learn the basic architecture of a dialogue system. Next, by looking at the change in behaviors after modifying the knowledge used in the sample applications, they understand elemental technologies. Then they deepen their understanding of the elemental technologies by building a new application using built-in blocks. Next, they understand the necessity of extensibility by creating and using their own custom blocks. Finally, by having people other than themselves use the system they built, they understand the importance of robustness in intention understanding and interaction design.

4 Usage Example of DialBB

DialBB was utilized in student projects and for system development for competitions. For instance, it was used to develop the system that won third place (Kubo et al., 2022) in the Dialogue Robot Competition 2022 (Minato et al., 2022) and the system that won second place (Yanagimoto et al., 2023) in the Dialogue Robot Competition 2023 (Minato et al., 2024). They are conversational robots that can recommend tourist destinations. DialBB applications work as their dialogue processing components. The dialogue processing component of the 2023 system incorporates the built-in Japanese Canonicalizer Block and the built-in STN Manager Block, along with a custom block that performs keyword-based language understanding, sentiment analysis, and affirmative/negative utterance classification.

5 Concluding Remarks

This paper presented DialBB, a framework for developing dialogue systems. DialBB serves as an educational material for dialogue system technology.

Currently, we are building a GUI-based editor for state-transition networks. Future improvements include adding new built-in blocks. Additionally, we plan to develop new sample applications, incorporating useful examples such as frame-based

dialogue management, database access, and handling speech and multimodal input/output.

We will demonstrate sample applications and explain their configuration files and knowledge for language understanding and dialogue management, to show how DialBB is useful in learning dialogue system technology.

Acknowledgements

We would like to thank those who used the earlier versions of DialBB and gave us useful feedback.

This work was partly supported by JSPS KAKENHI Grant Number JP22H00536.

References

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *Preprint*, arXiv:1712.05181.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Preprint*, arXiv:1805.10190.

Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. 1980. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys*, 12(2):213–253.

Yuki Kubo, Ryo Yanagimoto, Hayato Futase, Mikio Nakano, Zhaojie Luo, and Kazunori Komatani. 2022. Team OS's system for Dialogue Robot Competition 2022. In *Proc. Dialogue Robot Competition 2022*.

Akinobu Lee, Keiichiro Oura, and Keiichi Tokuda. 2013. MMDAgent–a fully open-source toolkit for voice interaction systems. In *Proc. ICASSP*, pages 8382–8385.

Pierre Lison and Casey Kennington. 2016. OpenDial: A toolkit for developing spoken dialogue systems with probabilistic rules. In *Proceedings of ACL-2016 System Demonstrations*, pages 67–72, Berlin, Germany. Association for Computational Linguistics.

Takashi Minato, Ryuichiro Higashinaka, Kurima Sakai, Tomo Funayama, Hiromitsu Nishizaki, and Takayuki Nagai. 2022. Overview of Dialogue Robot Competition 2022. In *Proc. Dialogue Robot Competition 2022*.

Takashi Minato, Ryuichiro Higashinaka, Kurima Sakai, Tomo Funayama, Hiromitsu Nishizaki, and Takayuki Nagai. 2024. Overview of Dialogue Robot Competition 2023. In *Proc. Dialogue Robot Competition 2023*.

Daniel Ortega, Dirk Väth, Gianna Weber, Lindsey Vanderlyn, Maximilian Schmidt, Moritz Völkel, Zorica Karacevic, and Ngoc Thang Vu. 2019. ADVISER: A dialog system framework for education & research. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–98, Florence, Italy. Association for Computational Linguistics.

Stefan Ultes, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gašić, and Steve Young. 2017. PyDial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78, Vancouver, Canada. Association for Computational Linguistics.

Ryo Yanagimoto, Yunosuke Kubo, Miki Oshio, Mikio Nakano, Kenta Yamamoto, and Kazunori Komatani. 2023. User-adaptive tourist information dialogue system with yes/no classifier and sentiment estimator. In *Proc. Dialogue Robot Competition 2023*.

Qi Zhu, Christian Geishauser, Hsien chin Lin, Carel van Niekerk, Baolin Peng, Zheng Zhang, Michael Heck, Nurul Lubis, Dazhen Wan, Xiaochen Zhu, Jianfeng Gao, Milica Gašić, and Minlie Huang. 2023. ConvLab-3: A flexible dialogue system toolkit based on a unified data format. *Preprint*, arXiv:2211.17148.