The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

我们造了个系统，很牛，就是不给你用

Chatti: A Conversational Chatbot Platform

Inchul Hwang, Heesik Jeon, Hyung Rai Oh, Donghyeon Lee, Munjo Kim, Jihie Kim

Artificial Intelligence Team, Samsung Electronics Co., Ltd., Seoul, Korea
{inc.hwang, heesik.jeon, hyungrai.oh, dh.semko.lee, munjo.kim, jihie.kim}@samsung.com

Abstract

We demonstrate the conversational Chatbot platform named Chatti which supports developers with a tool to develop their chatbot easily without full understanding technologies inside a conversational chatbot. To develop a chatbot with Chatti, a developer inputs customized domain data and deploys his Chatbot with a tool. Then users can interact with the Chatbot based on natural language conversation via messengers and so on.

Chatti includes natural language understanding, dialog management, action planning, natural language generation and chitchat component which run on g models learned from developers' input data as in common in conversational assistants such as Bixby, Siri, Alexa and etc. With Chatti, the developer could make his Chatbot support two types of conversation simultaneously – basic chitchat and task-oriented dialog. In contrast to prior chatbot building tools are mainly focused on the Natural Language Understanding, Chatti is more focused on full dialog system – dialog management, action planning, natural language generation and chitchat. We believe Chatti could accelerate a wide possibility of conversational Chatbot for services as well as IoT devices.

1. Introduction

In recent years, a Chatbot has been widely used as a communication channel with users via messengers – Facebook, skype, etc. To handle various requests from users based on natural language, a Chatbot should understand what an intention of a user is and what parameters expressed in a user's utterance. Additionally, a Chatbot should execute the right actions or the dialog connections based on dialog context. And then, a Chatbot should generate a natural response correctly based on the system result and dialog context. All of these technologies are very similar to a typical Dialog System – natural language understanding, dialog management, natural language generation as shown in Figure 1.

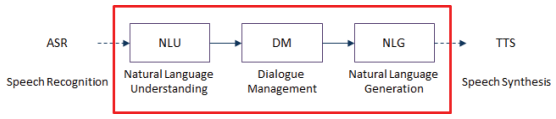

Figure 1 Typical Dialogue System Architecture

Based on dialog system, many companies have been investing in an intelligence personal assistant (Samsung Bixby, Apple Siri, Amazon Echo and etc.) as an interface. Because a conversational interface could be provided in various services including voice-based assistants for IoT devices, we developed the conversational Chatbot platform named Chatti, which was developed based on technologies in as personal assistant systems. With Chatti, developers could develop their own Chatbot very easily – makes their Chatbot understand what users want to do by doing the natural dialog connections. Chatti provides a well-designed integration of components of dialog systems and supports customization by data. Therefore, we believe that Chatti could accelerate a wide possibility of Chatbot for services as well as IoT devices when it needs a conversational interface. In this demonstration, we show how to use Chatti for the development of Chatbot and the sample Chatbot service.

In section 2 of this paper, we explain more details of Chatti. In section 3, we describe how to do the demonstration. We then summarize the paper and future work in section 4.

2. Chatti Architecture

The proposed system architecture is composed of the following components as shown in Figure 2. To handle user's interaction based on text, we developed 5 modules inside the Chatti as follows:

- Dialog Management (DM) to maintain dialog context and Action Planning (AP) to plan the right sequence of service actions. These modules could be a code in other

platforms. In Chatti, these modules are supported as a part of platform, so developers could easily utilize dialog management and action planning functionalities – e.g. dialog state tracking, dialog policy decision, planning based on the task network based on the conversation with a specific user.

- Natural Language Understanding (NLU) to extract intents and entities from user's utterance. There are bunch of approaches to implement NLU functionality – Pattern-based, Statistical Approach and Deep Neural Network (LSTM, CNN, etc..). In Chatti, there are various approaches (Dictionary, Pattern-based, an so on.)
- Natural Language Generation (NLG) to generate system response based on text form. NLG could also be implemented with various methods – simple sentence, template-based and DNN-based. We also support various ways which developers could utilize including simple sentence, template-based and so on.
- Chitchat to make the automatic chitchat response. Regarding chitchat if the dialog corpus is enough, we could do the End-to-end learning based on dialog corpus. Additionally, if corpus data is not enough, a method based on Information retrieval could be better rather than End-to-end learning approach.
- Knowledge Base (KB) to store and manage the data which is used for each module such as a model and corpus

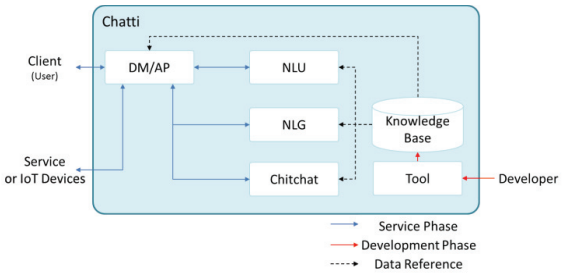

Figure 2 Typical Dialogue System Architecture

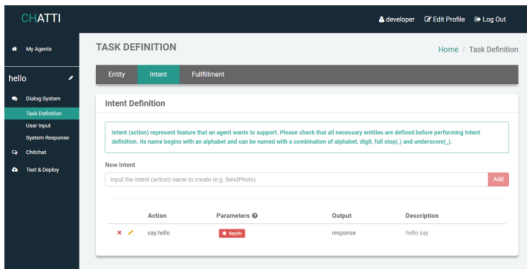

Figure 3 Snapshot of Chatti Development Tool

To build a customized Chatbot, the developers can use the tool in the Chatti in Figure 3.

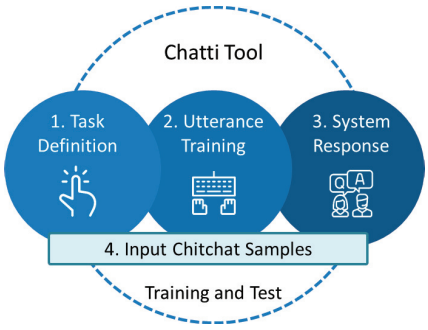

Figure 4 Overall 4 steps of Development

There are 4 steps for developers to build the Chatbot as shown in Figure 4 with Chatti tool(below example is supposed to be made for 'Weather.bot' which serves a weather service).

1. Defining intentions and entities including options for dialog policies of a Chatbot. Intentions describe the functionalities of a Chatbot and entities describe the input/output parameters of the Chatbot. For example, weather information search functionality could be described as a intention – 'search.weather', entities could be a location and a status of weather as an input/output parameters. When a developer describes entities, a developer could set some properties of entities such as if it could be multiplicity or mandatory for the intention. After defining all intentions and entities, defined tasks should be connected to the IoT devices or back-end services which describes with Open APIs. To make Chatti connect to IoT devices or back-end services, a developer should define Webhook which calls open APIs based on the request from Chatti and returns the results to Chatti. With defining Webhook URL into Chatti, developers could do this work. Developers could check the actual task network via tool after authoring an intention with graph type as in Figure 5.
2. Base on the intentions and entities defined in 1, next step is to input raw data such as utterance, dictionary and so on. After inputting the raw data, developers should tag them as a NLU corpus. For example, 'what's the weather in Seattle (Intention: 'search.weather', Entity-location: 'Seattle')' could be an example. Because the corpus data could be large, bulk import/export is supported. Additionally, a developer could test their trained NLU model on-the-fly based on input text or batch-style text file – system will show the confidence level and slot information on the tool.
3. Inputting system response based on dialog states, user's utterance including user's intentions/slots as well as a system result of execution. For example, if the searching weather is successfully done, the response could be 'Weather is [status]fine in [location]Seattle' – [status] is from the results of system execution and [location] is from the user's utterance).

4. Inputting sample utterances and responses pair for Chitchat. For example, if a user says 'what can you do?', then a response could be 'I'm a weather bot'. Then this pair 'what can you do?'-'I'm a weather bot' should be input to the system. If a user's utterance is similar to the input utterance data –'what can you do', the Chatbot will give the response 'I'm a weather bot'.

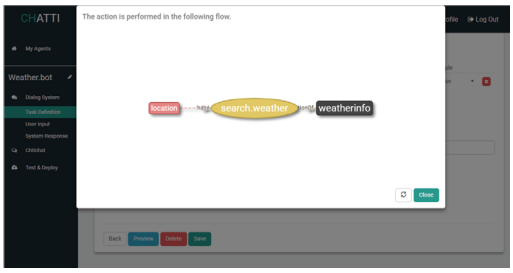

Figure 5 Task Network based on Graph View

After inputting all raw data, if a developer trains each module, the system will generate models for the module. As a result, after a developer finishes all of these phase, data in Knowledge base will be as follows:

- Task network which describes the functionalities (intentions) as well as parameters (entities)
- NLU corpus (raw data) and NLU trained model
- NLG template
- Chitchat trained model

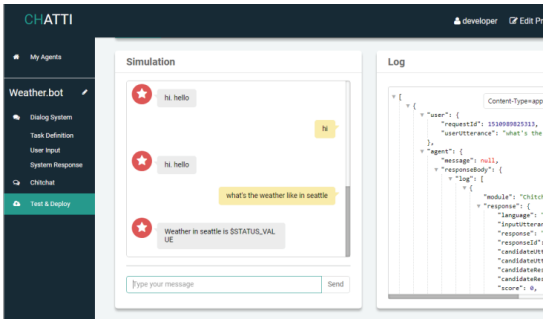

Figure 6 Testing Chatbot on Chatti Tool

Additionally, developers could test the whole Chatbot based on the model and raw data with or without Webhook on Chatti tools as shown in Figure 6. Like a messenger, developers could input any utterance and see the actual response from Chatbot.

The chatti system has been used in developing several chatbots: Meeting planner, Q&A on bus schedule, controlling washing machine, etc. It tasks several days(2~3 days) to develop these chatbots with 7 developers.

3. Demonstration

Our demonstration system consists of two parts, the tool for developers and sample Chatbot service as in Figure 7.

Figure 7 Environment for Chatti Demonstration

In the first, we demonstrate the authoring a Chatbot based on Chatti tool. In this phase, we are going to use our tool on a monitor or projector which displays tools and explain each step for authoring a specific Chatbot. – Defining the task, inputting NLU raw data, defining NLG and inputting chitchat data and testing developed one. We will give a simple example such as 'Weather.bot' as described in this paper.

In the second part, we demonstrate the sample bot which we developed as a sample - the laundry bot which handles a user's request for a laundry via messenger named Kakao talk which is the most popular in Korea. A smartphone will be connected to a monitor and another monitor will be shown the virtual laundry which is connected to the laundry bot with a Webhook. So, if a user sends a message regarding the laundry via Kakao talk, the Chatbot will give answers or do the actions with virtual Laundry bot.

4. Conclusion

In this demonstration, we show Chatti which provide the development environment with developers for efficient development of a Chatbot. With Chatti, developers could utilize Dialog System technologies without deep knowledge by just inputting domain specific data to the system with a tool. Additionally, testing including system level as well as component level is supported by Chatti, debugging and improving is also possible.

In the future, we are going to improve Chatti with various ways – improvement of dialog system technology such as End-to-end dialog modeling as well as functionality such as log analytics for a Chatbot.

References

Samsung Bixby, http://www.samsung.com/sec/apps/bixby/

Apple Siri, http://www.apple.com/ios/siri/

Amazon Alexa, http://alexa.amazon.com/spa/index.html

Hauswald, J., Laurenzano, M.A., Zhang, Y., Li, C., Rovinski, A., Khurana, A., Dreslinski, R.G., Mudge, T., Petrucci, V., Tang, L. and Mars, J., 2015, March. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In ACM SIGPLAN Notices (Vol. 50, No. 4, pp. 223-238). ACM.

Jeon, H., Oh, H.R., Hwang, I. and Kim, J., 2016, March. An Intelligent Dialogue Agent for the IoT Home. In AAAI Workshop: Artificial Intelligence Applied to Assistive Technologies and Smart Environments.

ARIA, http://aria-agent.eu/

Chen, Y.N., Celikyilmaz, A. and Hakkani-Tür, D., 2017. Deep Learning for Dialogue Systems. Proceedings of ACL 2017, Tutorial Abstracts, pp.8-14.

Henderson, M., 2015. Machine learning for dialog state tracking: A review. In Machine Learning in Spoken Language Processing Workshop.

Liu, B. and Lane, I., 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. arXiv preprint arXiv:1609.01454.

Wen, T.H., Gasic, M., Mrksic, N., Su, P.H., Vandyke, D. and Young, S., 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. arXiv preprint arXiv:1508.01745.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J. and Jurafsky, D., 2016. Deep reinforcement learning for dialogue generation. arXiv preprint arXiv:1606.01541.