On Design and Implementation a Federated Chat Service Framework in Social Network Applications

Shi-Cho Cha* Zhuo-Xun Li* Chuan-Yen Fan* Mila Tsai* Je-Yu Li* Tzu-Chia Huang*
*Dept. of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan

***Abstract*—As many organizations deploy their chatbots on social network applications to interact with their customers, a person may switch among different chatbots for different services. To reduce the switching cost, this study proposed the Federated Chat Service Framework. The framework maintains user profiles and historical behaviors. Instead of deploying chatbots, organizations follow the rules of the framework to provide chat services. Therefore, the framework can organize service requests with context information and responses to emulate the conversations between users and chat services. Consequently, the study can hopefully contribute to reducing the cost for a user to communicate with different chatbots.**

***Index Terms*—Chatbots, Chat Services, Messengers**

I. Introduction

The chatbot research originates from the ELIZA proposed by Weizenbaum in 1966 [1]. Recently, because the social network applications, such as Facebook, Twitter, Slack, Line, etc., have opened their application programming interfaces (APIs) [2], the chatbot technologies have been brought into the spotlight: Organizations can develop their chatbots based on the APIs and emulating conversation with users of the social network applications [3]. As the popularity of the social network applications, an organization can reach a large number of customers or potential customers.

Compared to traditional command-line user interface, the conversational interface of a chatbot enables users to refine their needs interactively [4]. Thanks to the advances of artificial intelligence (AI) and natural language processing (NLP) technologies, current chatbots can provide 24/7 human-like and customized services cost efficiently. Moreover, several chatbot building platforms, such as Google Dialogflow, Microsoft Bot Framework, Chatfuel, and other platforms [5], have been proposed to help organizations to establish their chatbots. Therefore, more and more organizations deploy their chatbots on the social network applications. For example, Facebook reported that there are over 40 million active business chatbots on Facebook Messenger in 2019 [6]. Consequently, a person may pay switching cost to use chatbots provided by different parties. For instance, when a person negotiates with a chatbot for a while to buy a cinema ticket, the person needs to start from scratch to negotiate with the other chatbot if the person decide not to buy the ticket with the former chatbot.

To reduce the switching cost, current researchers proposed the concept of personal chatbots [7] [8]. Personal chatbots can keep track of user behaviors. Therefore, the personal chatbots can use the collected data to assist users to request application services in personalized ways. However, to the best of our knowledge, current research does not address the issue of communicating with chatbots provided by different organizations. In light of this, this study proposes the Federated Chat Service Framework, which provides a unified conversational interface to users for utilizing services offered by different organizations in social network applications.

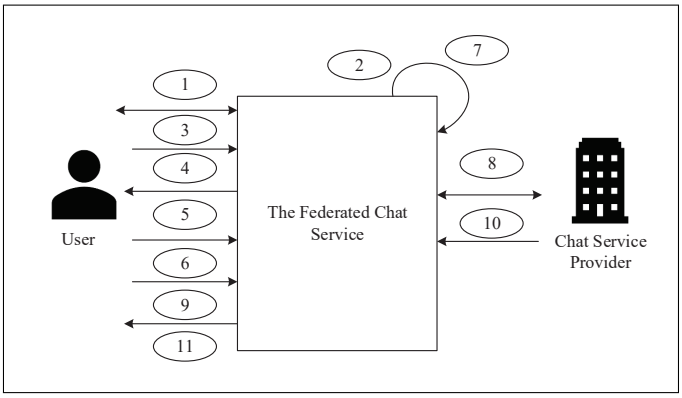

Fig. 1. The Scenario of Using the Proposed Framework

The rest of this paper is organized as follows: Section II provides an overview of the proposed framework. In the proposed framework, each organizations should provide their chat services. Therefore, this study describes the concept of chat services in Section III. Conclusions are finally drawn in Section V along with recommendations for future research.

II. Framework Overview

This study depicts the scenario of using the proposed framework in Fig. 1. In addition, this study uses Fig. 2 to illustrate the architecture of the framework. As shown in the figures, the kernel of the framework is the Federated Chat Service. The service plays the role of chatbots in social network applications and provides *interpreters* to convert messages from/to a social network application into standard messages for the service. Therefore, a user can communicate with the chatbots to use the service (Step 1 in Fig. 1). Note that this study does not specify who will provide the federated chat service. The federated chat service can be provided by a cloud service provider or a user himself/herself. This study leaves the deployment issue of the service to our future work.

The framework requests each chat service provider offers chat service with the specified *chat service interface*. A chat service provider then registers its service through the

service management interface of the federated service. The chat service provider can modify its configuration of its service via the service management interface later.

Before a user decides to use a chat service, the *Mediator* of the federated chat service requests the *Chat Moderator* to deals with messages from the user and transfer responses back to the user (Step 2 in Fig. 1). The chat moderator can be viewed as a general purposed intelligent agent interacting with the user. In the meanwhile, the mediator collects chat logs via the *Chat Log Manager* and requests the *Profile Manager* to retrieve user profiles from related social network applications. Furthermore, a user can link his/her accounts in different social network applications and set privacy preferences (Step 3 in Fig. 1).

During the conservation with the user, the moderator can further update user profiles based on chat logs. In addition, with the *service recommendation* component, the moderator may recommend the user to use a service based on user intents, profiles, and context information (Step 4 in Fig. 1). If the user determine to use the service provided by a chat service provider, the user can send a request to the federated chat service to start a conversion session with the service (Step 5 in Fig. 1). Then, when the federated chat service receives messages from the user (Step 6 in Fig. 1), the federated chat service invoke the related wrapper to send the messages to the service (Step 7 in Fig. 1). In this case, the *Session Manager* keeps session states of a communication session. Therefore, a user can suspend a session and resume it later. A chat service deals with user messages and provides responses back to the user via the federated chat service (Step 8 and 9 in Fig. 1). Moreover, a chat service may send notifications to a user in a proactive manner (Step 10 and 11 in Fig. 1). Finally, the user may decide to close the communication session or use another chat service without closing the session.

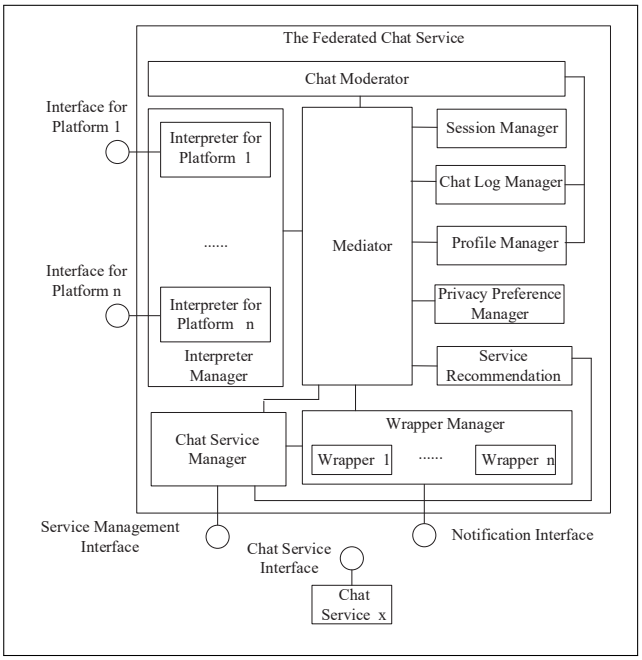

Fig. 2. The Architecture of the Proposed Framework

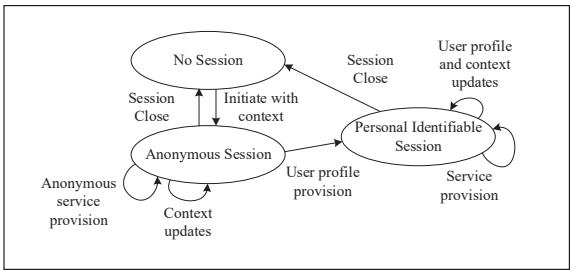

Fig. 3. The State Diagram of a Chat Service

III. Chat Services

Fig. 3 demonstrates the state diagram of a chat service. The concept of a chat service is related to communication sessions. Currently, the framework provides two types of communication sessions – the anonymous sessions and the personal identifiable sessions. Before the federated chat service can send messages of a user to a chat service, the federated chat service needs to initiate a new session or use an existing session of the user. When the federated chat service requests to initiate a session from a chat service, it provides context information so that the chat service can provide services based on the information. Moreover, the federated chat service can notify the chat service to update the context information when the user uses other chat services.

This study calls the session mentioned previously as an anonymous session because the chat service provider does not obtain user profiles or other personal identifiable information. If a user allows a chat service to obtain his/her profiles or previous chat logs, the federated chat service can provide the data to the chat service. Therefore, the chat service can provide personalized service based on the information or link the information to previous personal identifiable sessions. However, providing personal identifiable information invades forward privacy. This study leaves the limitation to our future work.

IV. Prototype Illustration

This study implements a simplified prototype to illustrate the proposed framework. The prototype focuses on the LINE platform. As depicted in Fig. 4, the prototype plays the role of a LINE bot and uses the LINE Messaging API to communicate with LINE users. This study implements the prototype and associated components with the Spring boot framework (version: 2.1.8.RELEASE) using JDK 8u221, Java Servlet 4.0.1 and Tomcat 9.0.24. Also, the prototype is implemented with the LINE Messaging API SDK for Java (version 3.0.0). A LINE user can add the bot of the prototype as his/her friend and communicate with the bot. Take Fig. 5 for example, users can send commands to the bot by putting the symbol # in front of the commands. For examples, users can send #Topic to list the top level topic types of registered chat services. The bot uses LINE buttons to list the results so that users can click a

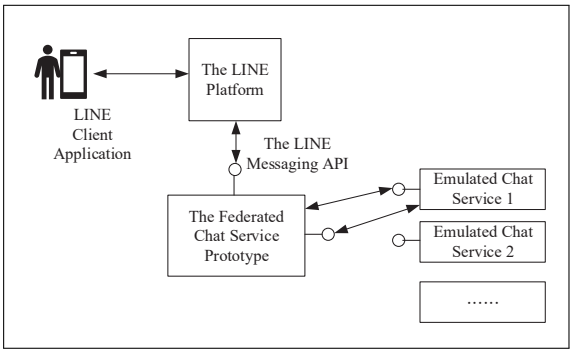

Fig. 4. The Prototype of the Proposed Framework

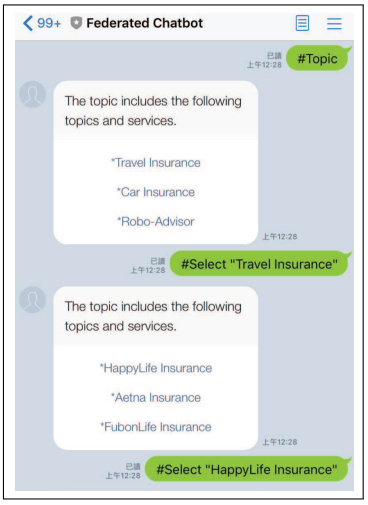

Fig. 5. Users can Talk with the Prototype and Express their Intents

specific button to list the sub-topics and chat services of the associated topic. A sub-topic may further include a set of sub-topics and a set of chat services. This study uses the symbol to differentiate a topic (or a sub-topic) from a chat service.

Instead of selecting a chat service with buttons, this study enable a user to select a service or a topic with the command *Select*. That is, a user can select a topic or a chat service with its name. If a user selects a topic directly, the bot lists sub-topics and chat services of the selected topic. Comparatively, if the user select a chat service, the bot enables the user to communicate with the selected chat service. Finally, a user can send the *help* command to request the bot to list commands or to request the bot to show the usage of a command.

Moreover, the current prototype enables users to manage messages to be shared with rich menus (Fig. 6). First, a user can use the Record command to request the bot to start recording requests and responses with chat services. When the bot is recording messages, a user can send the record command to the bot to stop recording. Then, a user can send the *List Record* command to view recorded messages and use the *Clear* command to clear recorded data.

Fig. 7 demonstrates the process for a user to communicate with a chat service. Upon a user deciding to have a dialogue

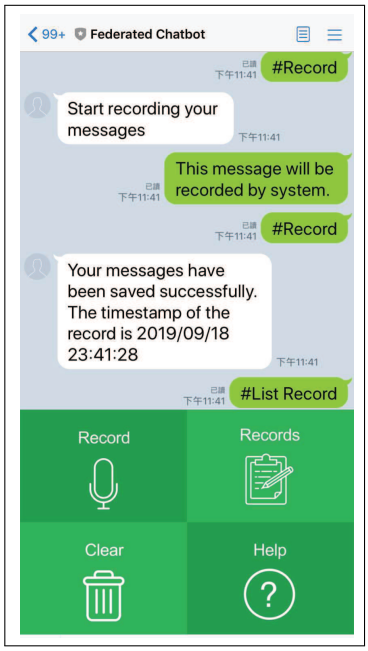

Fig. 6. Controls

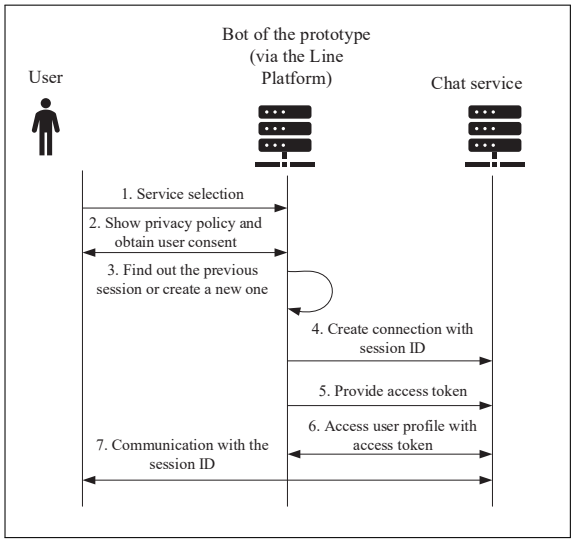

Fig. 7. The Sequential Diagram for a User to Communicate with a Chat Service.

with a chat service (Step 1 in Fig. 7), the bot of the prototype provides the privacy policy of the chat service to the user (Step 2 in Fig. 7). Then, as shown in Fig. 8, the bot obtains user consent by asking the user to answer the following two questions:

- Do you want to create a new session? If the user clicks the "yes" button, the bot will use the same session ID for the user to communicate with the service. Otherwise, it means that the user wishes to establish an anonymous session mentioned in Section III with the chat service. Therefore, the bot generates a new session ID every time

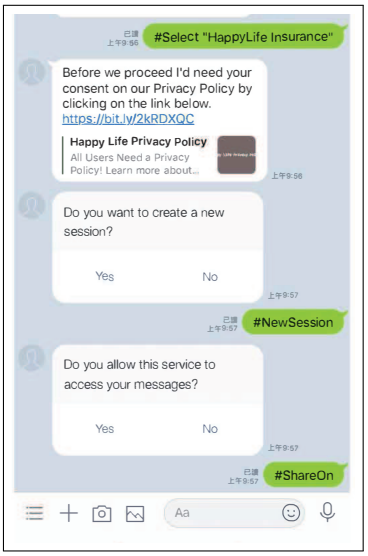

Fig. 8. Obtaining User Consent for the Chat Service.

when the user selects the chat service (Step 3 and Step 4 in Fig. 7).

- Do you allow this service to access your messages? If the user chooses "yes", the bot will authorize the chat service to access recorded messages of the user by providing an access token to the service (Step 5 and Step 6 in Fig. 7).

Finally, the user can use the session ID decided in Step 3 (Fig. 7) to communicate with the chat service.

V. Conclusions and Future Work

To reduce the switching cost of using different chatbots, this study proposed the Federated Chat Service Framework. The framework maintains user profiles and historical behaviors. Instead of deploying chatbots, organizations follow the rules of the framework to provide chat services. Therefore, the framework can organize service requests with context information and responses to emulate the conversations between users and chat services. Consequently, the study can hopefully contribute to reducing the cost for a user to communicate with different chatbots.

This study has implemented a prototype system to illustrate the concept of the proposed framework. There are many interesting things left to be done. First, the prototype only supports the LINE platform. Therefore, this study is going to identify a common interface for different chat platforms and to implement interpreters for the platform.

Second, the current prototype requests users to select chat services themselves. However, users may have trouble selecting a chat service from a lot of services. Developing an intelligent recommendation system would reduce the cost of service selection. Moreover, we can design and implement a system for chat services to compete with one another for the chance to communicate with a user. This study leaves the issue to our future work.

Third, the current prototype allows users to share chat logs with other chat services. In this case, users may classify logs into categories and share different log categories with different chat services. Providing a friendly user interface for users to classify their logs and to authorize chat services to access logs of specific categories is worth for future studies.

Last but not least, a chat service may not wish other chat services to obtain messages, such as special price offers, for users. It would be an interesting future work to discuss the legal or ethical issues for users to share their chat histories of a chat service with other chat service.

Acknowledgement

This work was supported in part by the Taiwan Ministry of Science and Technology under grants 105-2221-E-011-079-MY3 and 107-2218-E-011-012.

References

[1] J. Weizenbaum, "ELIZA – a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966.

[2] P. B. Brandtzaeg and A. Følstad, "Chatbots: Changing user needs and motivations," *Interactions*, vol. 25, no. 5, pp. 38–43, Aug. 2018.

[3] M. Vaziri, L. Mandel, A. Shinnar, J. Siméon, and M. Hirzel, "Generating chat bots from web api specifications," in *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. New York, NY, USA: ACM, 2017, pp. 44–57. [Online]. Available: http://doi.acm.org/10.1145/3133850.3133864

[4] R. Batish, *Voicebot and Chatbot Design: Flexible Conversational Interfaces with Amazon Alexa, Google Home, and Facebook Messenger*. Packt Publishing, 2018.

[5] A. Patil, K. Marimuthu, N. Rao, and R. Niranchana, "Comparative study of cloud platforms to develop a chatbot," *International Journal of Engineering & Technology*, vol. 6, no. 3, pp. 57–61, 2017.

[6] T. Helwick, "Messenger at f8 2019 — over 20b messages exchanged between people and businesses per month," Online Document., year = 2019.

[7] F. Daniel, M. Matera, V. Zaccaria, and A. Dell'Orto, "Toward truly personal chatbots: On the development of custom conversational assistants," in *Proceedings of the 1st International Workshop on Software Engineering for Cognitive Services*. New York, NY, USA: ACM, 2018, pp. 31–36.

[8] D. Rooein, "Data-driven edu chatbots," in *Companion Proceedings of The 2019 World Wide Web Conference*. New York, NY, USA: ACM, 2019, pp. 46–49.