Tuning a conversation strategy for interactive recommendations in a chatbot setting

Yuichiro Ikemoto\textsuperscript{a}, Varit Asawavetvutt\textsuperscript{a}, Kazuhiro Kuwabara\textsuperscript{b} and Hung-Hsuan Huang\textsuperscript{b}

\textsuperscript{a}Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga, Japan; \textsuperscript{b}College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga, Japan

\textbf{ABSTRACT}

This paper presents a conversation strategy for interactive recommendations using a chatbot. Chatbots have recently been attracting attention for their use as a flexible user interface. To develop an effective chatbot, it is important to determine what kind of questions to ask, what information should be provided, and how to process a user’s responses for a given task. In this paper, we target a chatbot that uses a graphical user interface (GUI) and focus on the task of recommending an item that suits a user’s preference. We propose a conversation strategy where a chatbot combines questions about a user’s preferences and recommendations while soliciting user’s feedback to them. The balance between the questions and recommendations is controlled by changing the parameter values. In addition, we propose a simulation model to evaluate the performance of interactive recommendation under different parameter values. The simulation results with a prototype dataset are presented and discussed.

\textbf{1. Introduction}

This paper presents a conversation strategy for a chatbot by focusing on interactive recommendation tasks. A chatbot generally exploits natural language processing so that it does not depend on a particular task or domain. Various frameworks have been proposed to facilitate the development of chatbots (e.g. Augello, Scriminaci, Gaglio, & Pilato, 2011; Yan, Castro, Cheng, & Ishakian, 2016). For a given task, not only natural language processing but also how to conduct conversations in a particular context is important. Some chatbots also rely on a graphical user interface (GUI), such as buttons with which the user can send a predefined message by just clicking on a button without manually inputting any text. In such a case, not only what kind of messages are sent to users but also what possible answers are provided to users are important.

In this paper, we focus on a task that recommends an item that matches a user’s preferences, and consider a conversation strategy in a chatbot setting. Typical recommender
systems use algorithms such as content-based or collaborative filtering (Ricci, Rokach, & Shapira, 2015). In these systems, the interaction between a system and a user is typically one-shot. However, in conversational recommender systems, a user repeatedly interacts with a system. For example, a critiquing-based recommender system uses the user’s feedback or critiques about recommended items to narrow down suitable items for recommendation (Chen & Pu, 2012). We apply methods proposed in recommender system research to the implementation of a chatbot. In doing so, we also employ GUI functions provided on some chatbot platforms for a better user interface.

This paper is a revised and extended version of our ACIIDS 2018 paper (Ikemoto, Asawavetvutt, Kuwabara, & Huang, 2018) in which we proposed a conversation strategy that control the balance between questions and recommendations by parameters, and preliminary evaluation results using the prototype we developed were reported. In evaluating the system, we set parameter values based on the results of preliminary ad hoc trials. However, it was not clear how to systematically set proper parameter values for a desirable conversation. In this revised paper, we propose a user model for simulating interactions between a system and a user, which provides the basis for properly determining parameter values for a given dataset of items for recommendations. More specifically, we create various user models from a given dataset and simulate a conversation between a system and a user. By conducting simulations under different parameter values, we can examine the effects of parameters without the need for real users and choose appropriate parameter values.

The main difference from our ACIIDS 2018 paper is that, in addition to the conversation strategy for interactive recommendation using a chatbot, this revised paper describes a method of determining proper parameter values that characterize the conversation strategies using the proposed simulation model (Section 5), and reports a case study of the proposed method using a larger dataset (Section 6).

The remainder of the paper is organized as follows. The next section describes related work, and Section 3 describes a model for interactive recommendations using a chatbot. Section 4 explains the prototype we built using the LINE messaging service. In Section 5, we present a simulation model and in Section 6, we report the results of simulation experiments using the dataset prepared for the prototype. Finally, Section 7 concludes the paper and describes some future work.

2. Related work

2.1. Recommender systems

Inferring user preferences for items correctly is crucial for effective recommendations (Shi, Larson, & Hanjalic, 2014). A user’s preferences are generally represented by a user–item matrix. When a user’s previous behaviours are not known beforehand, the so-called cold start problem needs to be addressed. To solve this problem, a framework for eliciting user preferences was proposed (Christakopoulou, Radlinski, & Hofmann, 2016) that identified questions to learn a new user’s preferences. Another method was proposed where a series of recommendations is made and the user’s preferences are subsequently updated to reflect the user’s feedback for recommended items (Zhao, Zhang, & Wang, 2013).
In a conversational recommender system, obtaining feedback can be categorized into two basic types: navigation by asking and navigation by proposing (Smyth & McGinty, 2003). These feedback strategies are analyzed in terms of user efforts and cost. From the viewpoint of user interaction, generalized linear search (GLS) was also proposed that minimizes the number of user interactions to discover items that match a user’s interests (Kveton & Berkovsky, 2015). Adapting interaction strategies was also proposed in conversational recommender systems (Mahmood & Ricci, 2009) that use reinforcement learning techniques. Hariri, Mobasher, and Burke (2015) used a multi-armed bandit strategy to model online learning of the user preference changes. Conversational recommendations were also applied to acquire the functional requirements of products (Widyantoro & Baizal, 2014), where an ontology structure is introduced and user preferences are explored through question and answer interactions that resemble those between professional sales people and customers.

In contrast to these previous studies, we focus on a mechanism that switches between questions and recommendations and we apply it to a chatbot setting for interactions between a user and a chatbot. In addition, we focus on a method of tuning the parameter values that control interactions by introducing a user model for systematic simulation of interactions between a user and a chatbot.

2.2. Word retrieval assistant

Systems have been proposed that help aphasia sufferers recall an item’s name (Arima, Kuroiwa, Horiuchi, & Furukawa, 2015; Kuwabara, Iwamae, Wada, Huang, & Takenaka, 2016). Such word-finding difficulty is one typical symptom of a person with aphasia who has a clear mental image of an item but cannot find its name or the language to express it. A human conversation partner usually asks a series of questions to infer the name of the thing an aphasia sufferer wants to say. For example, suppose that an aphasia sufferer wants to say apple but cannot recall its name. A conversation partner asks such questions as Is it food?, Is it a fruit?, and Is it red?. The questions used in this setting are basically multiple choice or yes–no types of queries. If a person with aphasia answers the questions with a yes, what he wants to say might eventually be inferred as banana. The word retrieval assistance system asks a series of questions instead of a human conversation partner to infer what an aphasia sufferer is struggling to recall.

This word-finding process resembles a method that identifies an item whose characteristics a user knows but not the name of the item itself. In this sense, a word retrieval assistance system shares properties with interactive recommendations.

In word retrieval assistance systems, the order in which the questions are posed is critical to efficiently infer what the user has in mind. Typical heuristics calculate the information gain of questions and the question with the biggest information gain is asked next. A similar kind of heuristic will be required for an efficient interactive recommendation.

In addition, since the target user of a word retrieval assistance system has difficulty answering a question in a free-text form, a GUI, such as buttons, is heavily deployed. When a user answers a multiple choice question, using a GUI is more convenient than a free-text entry although the system needs to provide answer choices in addition to
questions. When GUI functions are available on a chatbot platform, they should also be exploited for interactive recommendations. In this paper, we assume the interactions that utilizes such GUI functions, and we focus on the conversation strategy suitable for such kinds of interactions.

3. Interactive recommendations in a chatbot setting

3.1. Recommendation model

We assume \( n \) items from which an item(s) is recommended to a user based on the user’s preferences. Each item is characterized by \( m \) properties. Item \( s_i \) (\( 1 \leq i \leq n \)) is represented by \( m \)-dimensional vector \( s_i = (s_{i1}, \ldots, s_{im}) \). Similarly, user \( u \)'s preferences are represented as \( m \)-dimensional vector \( \bar{u} = (u_1, \ldots, u_m) \), which we call user vector.

The similarity between user \( u \) and item \( s_i \), \( \text{sim}(u, s_i) \) is calculated based on the Pearson correlation coefficient:

\[
\text{sim}(u, s_i) = \frac{\sum_{j=1}^{m} (u_j - \bar{u})(s_{ij} - \bar{s}_i)}{\sqrt{\sum_{j=1}^{m} (u_j - \bar{u})^2} \sqrt{\sum_{j=1}^{m} (s_{ij} - \bar{s}_i)^2}}
\]

where \( \bar{u} \) designates the average value of \( u_j \) (\( 1 \leq j \leq m \)), and \( \bar{s}_i \) designates the average value of \( s_{ij} \) (\( 1 \leq j \leq m \)) for given \( i \).

3.2. Interactive recommendation flow

Figure 1 shows the overall interactive recommendation flow, which is started by a user who inputs a certain text such as Start. First, the user vector is initialized to \((0,0,\ldots,0)\); that is, the user’s preference is unknown at the start. Through interactions between a user and a system, the user vector’s value is updated. Here, we assume the value of \( s_{ij} \) is selected from a set of \((-2, -1, 0, 1, 2)\), and \(|u_j| \leq 2\).

![Figure 1. Overall interactive recommendation flow.](image-url)
Next, the similarity between an item and a user is calculated using formula (1) for all the items. If an item’s similarity exceeds the *recommendation threshold*, the item with the highest similarity is presented as a recommendation. This recommendation threshold is initially set to a parameter $\alpha$, which we call *initial recommendation threshold*. A bigger value of $\alpha$ means that a recommendation is only made when an item has been found that strongly matches the user’s preference. Thus, more questions will be asked before the first recommendation is made. In this way, we can manipulate the balance between the questions and recommendations by changing the value of $\alpha$.

As one heuristic, we decrease the recommendation threshold each time an additional recommendation is made to increase the chance of recommendations. How much the threshold value is decreased is controlled by another parameter called the *recommendation threshold decrement* $\gamma$ ($\gamma \leq 1$). Initially, the recommendation threshold is set to $\alpha$, and the subsequent threshold value is set by multiplying the previous threshold value by $\gamma$.

When there is no recommendation, we select a property to ask a user about. Here, the order in which questions are asked is important. For example, if all the items that could be recommended have the same value for a certain property, asking a question about that property is somewhat ineffective. Therefore, we calculate a set of items that might be recommended (more specifically, whose similarity value is not negative), and we calculate information entropy of each property. Since the item’s property value is selected from a set of $\{-2, -1, 0, 1, 2\}$, we can calculate the probability of a property $i$ ($1 \leq i \leq m$) having the value $v$ ($v \in \{-2, -1, 0, 1, 2\}$) assuming that items that might be recommended are selected with an equal probability. Based on this probability, information entropy can be calculated for each property $i$.

Next, a question about a property with the highest information entropy will be asked. If there are multiple properties with the highest information entropy, we will select one at random. In this way, we expect that the number of questions to be asked before narrowing down the items for recommendation can be reduced.

Depending on the recommendation threshold and the item with the highest similarity, a user is presented with either a question or a recommendation. Thus, we consider the following two types of interactions to update the user vector:

- *question*: a user is asked whether the user is interested in a particular property.
- *recommendation*: a user is presented with a recommendation and is asked whether the user likes it or not.

In the former case, the value of $u_j$ is updated according to the user’s answer to the question. Suppose that the possible responses are YES, NO, or NOT SURE. If the user’s response is YES, the value is increased by 1. If the updated value exceeds 2, it is set to 2. If the user’s answer is NO, the value is decreased by 1. If the updated value falls below $-2$, the value is set to $-2$. If the user’s response is NOT SURE, the value is not changed.

In the latter case, the value of $u_j$ is updated based on the user’s feedback in response to the recommendation. We categorize the possible feedback from the user as LIKE (or asking more for similar items), DISLIKE (or asking more for different items), or SKIP. If the feedback is LIKE, the property value of the user vector is increased if the same property
of the recommended item has a positive value. If the feedback is DISLIKE, the property value of the user vector is decreased if the same property of the recommended item has a positive value. Here, we only consider positive property values of items when we update the user vector, because they best characterize the target item and they may be worth reflecting onto the user vector. If the user’s feedback is NOT SURE, the user vector is not changed.

More specifically, for recommended item \( s_i \), the user vector is updated as follows:

\[
\begin{align*}
    u'_j & \leftarrow u_j + s_{i,j} \times \beta \\
    u'_j & \leftarrow u_j - s_{i,j} \times \beta \\
    u_j & \leftarrow \min \left( \max \left( u'_j, -2 \right), 2 \right)
\end{align*}
\]

Here, user feedback weight (\( \beta \)) determines how much a user’s feedback to a recommendation affects user vector (user’s preferences). Note that the value of user vector element is limited to the range of \(-2 \) and \(2 \) as shown in formula (3).

4. Prototype system

4.1. Dataset

We compiled a dataset of sightseeing spots in Kochi prefecture in Japan for our prototype. The number of sightseeing spots in the dataset is 102, and the number of properties is 25. We assigned an integer value from a set of \(\{-2, -1, 0, 1, 2\}\) to each property of a sightseeing spot manually. If the sightseeing spot has a strong tendency about the property, the value set is 2, and if it has a conflicting tendency about the property, the value set is \(-2 \).

Table 1 shows part of the data used in the prototype. In addition to the property values of the sightseeing spots, URLs of thumbnail pictures and related web sites are included in the dataset, and used in the GUI of the prototype.

4.2. System design

We used the LINE messaging service\(^1\) as a platform to construct a prototype chatbot system. The LINE platform provides a messaging API to facilitate the development of a chatbot. When a message is sent to the chatbot, a registered Webhook is invoked whose return value specifies the message that is to be returned to a user. In this prototype, we used Node.js\(^2\) to build a web server for Webhook, which is deployed on the Heroku cloud application platform (Figure 2).

| Property          | Nature | History | Temple | Castle | ... | Museum |
|-------------------|--------|---------|--------|--------|-----|--------|
| Sightseeing Spots |        |         |        |        |     |        |
| Kochi Castle      | \( -1 \) | 1       | \( -1 \) | 2      |     | 0      |
| Chikurin Temple   | 0      | 1       | 2      | \( -2 \) | ... | 0      |
| Ohtaru Falls      | 2      | \( -1 \) | \( -2 \) | \( -2 \) | ... | 0      |
| ⋮                 |        |         |        |        |     |        |
The LINE messaging API allows us not only to send messages but also to use GUIs like buttons or links to websites. In the prototype, we used buttons to let a user input the user’s responses. Figure 3(a) shows an example screenshot of asking a question to a user, and Figure 3(b) shows a recommendation of a sightseeing spot to a user and a request for feedback about it. When a recommendation is made, its thumbnail picture and a link to a relevant website are also shown to provide information on a recommended sightseeing spot.

5. Simulation model

To deploy an interactive recommendation system using the proposed conversation strategy, parameters such as $\alpha$, $\beta$, and $\gamma$ need to be set properly for effective interactions. Appropriate values depend on the dataset of target items and the system’s operation policy such as preferring recommendations over questions. As it is not easy to determine the best parameter values that are applicable universally, we create a model to simulate interactions between a user and the system. Using the simulation results, more appropriate parameter values can be determined for a given dataset.

Figure 4 shows the overall simulation process. We create a virtual user and let the system interact with it. In the proposed interactive recommendation, a virtual user needs (1) to respond to a question with YES, NO, or NOT SURE, and (2) to respond to a recommendation.

Figure 2. Overview of the prototype system.

Figure 3. Chatbot user interface. (a) Question and (b) Recommendation
with either LIKE, DISLIKE or SKIP. The user model needs to provide criteria of how to respond to a question and/or a recommendation. Here, we define a user model as a vector representing a user’s preference. As for replying to a question, if a corresponding property of a user vector has a positive value, YES will be selected as a response; if the property value is negative, NO will be selected; otherwise NOT SURE will be selected.

To determine a response to the recommended item, the similarity between an item and a user vector is calculated. The threshold values for answering LIKE and DISLIKE are determined beforehand for each user model. If the similarity between the user vector and a recommended item exceeds the former threshold, LIKE will be selected as a response; if the similarity falls below the latter threshold, DISLIKE will be selected; otherwise, SKIP will be selected.

One simulation run ends when there are no more questions to be asked (we do not ask the same question twice) or when all the items that exceed the recommendation threshold have been recommended. We record the questions asked and recommendations made to evaluate the performance of the interactive recommendation process.

6. Simulation results and discussion

We conducted simulation experiments to determine parameter values of the interactive recommendation process. We used the dataset of sightseeing spots that were created for the prototype system. These experiments are intended to show a case study of applying the proposed simulation method to determining the parameter values that control the conversation strategy. We focus on how the simulation results can be utilized to determine the parameter values rather than proving a certain hypothesis about the proposed recommendation algorithm.

6.1. User model

As mentioned in Section 4.1, the dataset prepared for the prototype contains 102 sightseeing spots, which are characterized by 25 properties. To simulate a variety of user models, we created 10 user models. First, we divided the sightseeing spots into 10 clusters using the \( k \)-means algorithm. We used Python machine learning library \textit{scikit-learn} to obtain the
clusters. Then, the centre of each cluster was calculated and used as a user vector of each user model. Figure 5 shows the sightseeing spot clusters and their centre using the t-SNE visualization method.

To determine the threshold of giving the feedback of LIKE (which we call user-item threshold), we examined the number of items (sightseeing spots) that are regarded as correct as the user-item threshold is varied. Figure 6 shows the results. Since the number

Figure 5. Clustering of sightseeing spots.

Figure 6. Number of correct items (sightseeing spots).
of correct items varies for a different user model, we plotted the maximum, minimum, and average of the number of correct items for all 10 user models.

To make the average percentage of correct items around 30%, the user-item threshold is set to 0.7 in the following simulation experiments. For the sake of simplicity, we adopted the same value of the user-item threshold for all the user models. This user-item threshold is used to determine if the user should answer with LIKE to a recommended item. For the threshold of answering NOT LIKE to a recommended item, we used 0 for all the user models.

In the following, we describe how various parameter values (initial recommendation threshold \( \alpha \), user feedback weight \( \beta \), recommendation threshold decrement \( \gamma \)) were determined in the proposed interactive recommendation system using the proposed simulation method.

6.2. Initial recommendation threshold \( \alpha \)

The recommendation threshold determines whether a recommendation is to be made or not, and it is set to the value of \( \alpha \) initially. To investigate the effects of the initial recommendation threshold \( \alpha \), which determines the balance between the questions and recommendations, we conducted a simulation using the user models described above, and counted the number of questions asked before the first recommendation was made for each user model. Figure 7 shows the results, where the maximum, minimum, and average number of questions asked before the first recommendation are plotted when the initial recommendation threshold \( \alpha \) is varied. Note that, for simplicity, when no recommendation is made, the number of questions asked is considered to be 26. This value is obtained by adding 1

![Figure 7. Number of questions asked before the first recommendation.](image-url)
to the maximum possible number of questions, which is equal to the number of properties characterizing items. Also note that, in these experiments, $\beta$ and $\gamma$ are irrelevant since only interactions before the first recommendation are considered.

Based on these results, we determined the initial recommendation threshold ($\alpha$) to be 0.5 in the following experiments since the average number of questions asked before the first recommendation is 4, which is considered appropriate for most users. Notably, the average number of questions asked before the first recommendation significantly increases when $\alpha$ is changed from 0.5 to 0.6.

6.3. User feedback weight ($\beta$)

The extent to which the user’s feedback on the recommended item is taken into consideration to update the user vector is determined by the user feedback weight ($\beta$). Intuitively, if we set the value of $\beta$ properly, the items that suit a user will more likely be recommended. We examined precision, which is the ratio of the correct items recommended to all the items recommended. Here, a correct item refers to an item whose similarity with the target user exceeds the user-item threshold.

Figure 8 shows the maximum, minimum, and average precision for all 10 user models when $\beta$ is varied. In this simulation experiment, $\gamma$ is set to 1 meaning that the recommendation threshold does not change. Big differences were observed in the results of different user models, and we could not determine the best $\beta$ value, which means that the feedback mechanism is somewhat unstable. However, since the average of precision is the highest with $\beta = 0.1$, $\beta$ is set to 0.1 in the following simulation experiments.

![Figure 8. Precision vs. user feedback weight ($\beta$).](image-url)
The recommendation threshold decrement parameter \( \gamma \) is meant to increase the chance of recommendations to be made, which is the number of items whose similarity value is higher than the threshold value, when additional recommendations are requested by a user. Each time the recommendation is made, the recommendation threshold used is updated by multiplying \( \gamma \) and the previous recommendation threshold. Thus, after \( n \) recommendations are made, the recommendation threshold becomes \( \alpha \times \gamma^n \). If \( \gamma \) is 1, the recommendation threshold does not change.

To investigate the effectiveness of \( \gamma \), we measured the precision and recall of correct items being recommended with different \( \gamma \) values. The value of \( \gamma \) is set so that after \( n \) recommendations, the recommendation threshold becomes \( 0.9 \times \alpha \) (initial recommendation threshold). We chose \( n \) to be either 3, 5, 10, or 100. The corresponding values of \( \gamma \) are 0.9645, 0.9791, 0.9895, and 0.9989. As mentioned in Section 6.1, we set the value of user-item threshold to 0.7, the initial recommendation threshold (\( \alpha \)) to 0.5 and the user feedback weight \( \beta \) to 0.1.

Figure 9 shows the maximum, minimum, and average precision and recall when \( \gamma \) is changed. As can be seen in Figure 9, when \( \gamma \) is less than 1, recall is improved compared with when \( \gamma = 1 \). Additionally, precision decreases as \( \gamma \) decreases.

For comparison, another simulation experiment was performed with the same conditions, except that the user-item threshold was set to 0.4 to increase the number of correct items. Figure 10 shows the maximum, minimum, and average precision and recall values according to changing \( \gamma \) when the user-item threshold is set to 0.4. With a larger number of correct items, the increase in recall is more apparent as \( \gamma \) decreases.
6.5. Discussion

User models for the simulation experiments were created from the given dataset of items using the clustering method. In this way, any number of user models can be created mechanically, and the performance of the interactive recommendation under a variety of user models can be examined.

Using this simulation model, we simulated interactions between a user and system with different parameter values using a dataset of our prototype as a case study. Although different user models produced different results, the general trends of the effects of parameter values can be known. For example, the initial recommendation threshold \( \alpha \) can control the balance between the questions and recommendations, and the recommendation threshold decrement \( \gamma \) can control the balance between the correctness and coverage of recommended items.

In this way, the simulation results can provide the data that serves as the basis for determining the value of parameters in interactive recommendation. This will facilitate the deployment of a recommender system.

7. Conclusion and future work

This paper described a conversation strategy that intersperses recommendations with questions for user preferences. The proposed conversation strategy can be controlled by setting parameter values. To determine proper parameter values, we proposed a simulation model with performance metrics. Using the dataset prepared for the prototype, we created user models and examined the recommendation performance. For the dataset used, we showed that the balance between the questions and recommendations can

![Figure 10. Precision and recall vs. recommendation threshold decrement (\( \gamma \)) (user-item threshold: 0.4).](image-url)
be controlled by changing the initial recommendation threshold ($\alpha$). The best value of $\alpha$ would be diverse for different datasets. Thus, the simulation model is useful for determining a proper value of $\alpha$ for a given dataset.

In addition, we varied the values of the user feedback weight ($\beta$) and found that the average precision varies little. This finding indicates room for improvement in the feedback mechanism. For example, we may need to ask a user the reason why the user likes or dislikes a particular sightseeing spot recommended. As for the recommendation threshold decrement ($\gamma$) for adjusting the recommendation threshold, the balance between the exactness and converge of recommended items can be controlled by the value of $\gamma$.

In the proposed simulation model, we first determined the correct sightseeing spots for each user model and examined the precision and recall of the correct sightseeing spots being recommended. In doing so, each user model is associated with a user-item threshold to determine which sightseeing spot is the correct one. In our simulation experiments, this threshold value was set to the same for all the user models. We may need to use a different threshold value for a different user model so that we can simulate a variety of users.

We might also need to consider performance metrics other than precision and recall. Since the system is assumed to be used interactively, the user’s subjective impression of using the system is important. For example, it might be more irritating for a user when unrelated questions are asked consecutively, even when the same number of questions is asked. Future work could also incorporate subjective elements into the simulation model.

**Notes**

1. https://line.me/en/
2. https://www.heroku.com/
3. http://scikit-learn.org

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

This work was partially supported by JSPS (Japan Society for the Promotion of Science (JP)) KAKENHI Grant Number 18K11451.

**Notes on contributors**

Yuichiro Ikemoto received his B.Eng. degree from Ritsumeikan University in 2017. He is currently a master’s course student at Ritsumeikan University. His research interests include a recommender system.

Varit Asawavetvutt received his B.Sc. degree from Thai-Nichi Institute of Technology in 2015, and M.Eng. from Ritsumeikan University in 2017. He was engaged in research on a chatbot system while he was a graduate student.
Kazuhiro Kuwabara received his B.Eng., M.Eng., and Dr.Eng. degrees from the University of Tokyo, in 1982, 1984, and 1997 respectively. In 1984, he joined Nippon Telegraph and Telephone Public Corporation (NTT), and was engaged in research and development on knowledge-based systems, multi-agent systems and socialware. He was with ATR Intelligent Robotics and Communication Laboratories from 2003 to 2006. Currently he is a professor at the College of Information Science and Engineering, Ritsumeikan University.

Hung-Hsuan Huang is a research scientist of Center of Advanced Intelligence Project, RIKEN, Japan since 2018. He received his B.S. and M.E. from National Cheng-Chi University and National Taiwan University in 1998 and 2000, respectively. In 2009, he received his PhD from the Graduate School of Informatics, Kyoto University, Japan. He then worked as a postdoctoral researcher at the Department of Computer and Information Science, Seikei University until he transferred to College of Information Science and Engineering, Ritsumeikan University as an assistant professor in 2010. He was promoted to be an associate professor in 2013 and held the position until transferring to the current one. His research interests are on applied artificial intelligence and human-computer interaction fields. Currently he intensively focused his works on embodied conversational agents (ECAs) and multi-modal interaction. He is the members of ACM, JSAI, IPSJ, HIS, IEICE, VRSJ, and TAAI.

References

Arima, S., Kuroiwa, S., Horiuchi, Y., & Furukawa, D. (2015). Question-asking strategy for people with aphasia to remember food names. Journal on Technology & Persons with Disabilities, 3, 10–19.

Augello, A., Scriminaci, M., Gaglio, S., & Pilato, G. (2011). A modular framework for versatile conversational agent building. 2011 international conference on complex, intelligent and software intensive systems (CISIS) (pp. 577–582). Seoul, Korea.

Chen, L., & Pu, P. (2012). Critiquing-based recommenders: Survey and emerging trends. User Modeling and User-Adapted Interaction, 22(1), 125–150.

Christakopoulou, K., Radlinski, F., & Hofmann, K. (2016). Towards conversational recommender systems. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 815–824). New York, NY, USA: ACM.

Hariri, N., Mobasher, B., & Burke, R. (2015). Adapting to user preference changes in interactive recommendation. Proceedings of the 24th international conference on artificial intelligence (pp. 4268–4274). Buenos Aires, Argentina: AAAI Press.

Ikemoto, Y., Asawavetvutt, V., Kuwabara, K., & Huang, H. H. (2018). Conversation strategy of a chatbot for interactive recommendations. In N. T. Nguyen, D. H. Hoang, T. P. Hong, H. Phamand, & B. Trawiński (Eds.), Intelligent information and database systems (pp. 117–126). Cham: Springer International Publishing.

Kuwabara, K., Iwamae, T., Wada, Y., Huang, H. H., & Takenaka, K. (2016). Toward a conversation partner agent for people with aphasia: Assisting word retrieval. In I. Czarnowski, M. A. Caballero, J. R. Howlettand, & C. L. Jain (Eds.), Intelligent decision technologies 2016: Proceedings of the 8th KES international conference on intelligent decision technologies (KES-IDT 2016) – part I (pp. 203–213). Cham: Springer International Publishing.

Kveton, B., & Berkovsky, S. (2015). Minimal interaction search in recommender systems. Proceedings of the 20th international conference on intelligent user interfaces (IUI 2015) (pp. 236–246). Atlanta, GA.

Mahmood, T., & Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. Proceedings of the 20th ACM conference on hypertext and hypermedia (pp. 73–82). New York, NY, USA: ACM.

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems handbook. Boston, MA: Springer.

Shi, Y., Larson, M., & Hanjalic, A. (2014, May). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. ACM Computing Surveys, 47(1), 3:1–3:45.

Smyth, B., & McGinty, L. (2003). An analysis of feedback strategies in conversational recommenders. The 14th Irish conference on artificial intelligence & cognitive science (AICS 2003) (pp. 211–216). Dublin, Ireland.
Widyantoro, D. H., & Baizal, Z. (2014). A framework of conversational recommender system based on user functional requirements. *2014 2nd international conference on information and communication technology (ICoICT)* (pp. 160–165). Bandung, Indonesia.

Yan, M., Castro, P., Cheng, P., & Ishakian, V. (2016). Building a chatbot with serverless computing. *Proceedings of the 1st international workshop on mashups of things and APIs* (pp. 5:1–5:4). New York, NY, USA: ACM.

Zhao, X., Zhang, W., & Wang, J. (2013). Interactive collaborative filtering. *Proceedings of the 22nd ACM international conference on information & knowledge management* (pp. 1411–1420). New York, NY, USA: ACM.