Conversation Strategy of a Chatbot for Interactive Recommendations

Yuichiro Ikemoto[1(✉)], Varit Asawavetvutt[1], Kazuhiro Kuwabara[2], and Hung-Hsuan Huang[2]

[1] Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu, Japan
is0201hh@ed.ritsumei.ac.jp

[2] College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan

Abstract. This paper presents a conversation strategy for interactive recommendations using a chatbot. Chatbots are attracting attention to provide a flexible user interface using natural language for various domains. For a given task, what kind of questions to ask and/or what information should be provided and how to process user responses play a crucial role in developing an effective chatbot. In this paper, we focus on a task of recommending an item that suits a user's preference and propose a conversation strategy where a chatbot combines questions about user's preference and recommendations soliciting user's feedback to them. The balance between questions and recommendations is controlled by changing the parameter values. We target a chatbot that uses a graphical user interface (GUI) and apply approaches proposed in the field of recommendation systems. Preliminary experiment results with a prototype indicate the potential of our proposed approach.

Keywords: Chatbot · Conversation strategy
Interactive recommendation

1 Introduction

This paper presents a conversation strategy for a chatbot by focusing on interactive recommendation tasks. Generally, a chatbot exploits natural language processing, which does not depend on a particular task or domain. Various frameworks have been proposed to facilitate the development of chatbots (for example, [2,12]). For a given task, not only natural language processing but also how to conduct conversations is important. Some chatbots also rely on such a graphical user interface (GUI) as buttons with which a user can send a predefined message by just clicking a button. In such a case, issues include not only what kind of messages are sent to users but also what possible answers are provided to them.

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.

N. T. Nguyen et al. (Eds.): ACIIDS 2018, LNAI 10751, pp. 117–126, 2018.
https://doi.org/10.1007/978-3-319-75417-8_11

Typical recommender systems use algorithms such as content-based or collaborative filtering [8]. In these systems, the interaction between a system and a user is generally one-shot. On the other hand, in conversational recommender systems, a user repeatedly interacts with a system. For example, critiquing-based recommender systems use user's feedback or *critiques* about recommended items to narrow down suitable items to recommend [3]. Here we apply such methods that have been proposed in recommender systems research to a chatbot setting. We also employ a GUI that is provided on some chatbot platforms to develop a more user-friendly system.

The rest of the paper is organized as follows. The next section describes related work, and Sect. 3 describes a model for interactive recommendations using a chatbot. Section 4 explains the prototype we built using the LINE messaging service, and Sect. 5 presents preliminary evaluation results obtained using our prototype. Finally, Sect. 6 concludes the paper and describes some future work.

2 Related Work

2.1 Recommender Systems

To achieve effective recommendations, user preferences for items must be inferred correctly [9]. A user's preferences are generally represented by a user–item matrix. When a user's previous behaviors 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 [4] 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 constantly updated to reflect user feedback on recommended items [13].

In a conversational recommender system, obtaining feedback can be categorized into two basic types: *navigation by asking* and *navigation by proposing* [10]. 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 [6]. Adapting interaction strategies was also proposed in conversational recommender systems [7] that use reinforcement learning techniques. Conversational recommendations are also applied to acquire the functional requirements of products [11]. This proposed framework introduces an ontology structure and explores user preferences through question and answer interactions that resemble those between professional sales people and customers.

In contrast to these studies, we focus on a mechanism that switches between questions and recommendations and apply it to a chatbot setting for natural interaction between a user and a chatbot.

2.2 Word Retrieval Assistant

Systems have been proposed that help aphasia sufferers recall an item's name [1,5]. Such word-finding difficulty is one typical symptom of a person with aphasia and

describes the plight of a person with aphasia who has a clear mental image of an item but cannot find its name or language to express it. In an activity that mirrors the traditional game of 20 questions, 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 *banana* but cannot recall its name. A conversation partner asks such questions as *Is it food?*, *Is it a fruit?*, *Is it yellow?*. The questions used in this setting are basically multiple choice or yes-no type of queries. If a person with aphasia answers them with *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 remember.

This word-finding process resembles a method that identifies an item whose characteristics a user knows but not 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 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. In a word retrieval assistance system, there is only one correct answer, but in a recommendation system, multiple items might be suggested. Thus, different heuristics are necessary for an 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 graphical user interface (GUI) is deployed, such as buttons. When a user answers a multiple choice question, using GUI is more convenient than a free-text entry. When GUI elements are available on a chatbot platform, they should also be exploited for interactive recommendations.

3 Interactive Recommendations with a Chatbot

3.1 Recommendation Model

We assume n items from which an item(s) is recommended to a user, based on her preferences. Each item is characterized by m *properties*. Item s_i is represented by m-dimensional vector $\boldsymbol{s_i} = (s_{i,1}, \cdots, s_{i,m})$. Similarly, user u's preferences are represented as m-dimensional vector $\boldsymbol{u} = (u_1, \cdots, u_m)$.

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

$$\mathrm{sim}(u, s_i) = \frac{\sum_{j=1}^{m}(u_j - \overline{u_j})(s_{i,j} - \overline{s_{i,j}})}{\sqrt{\sum_{j=1}^{m}(u_j - \overline{u_j})^2}\sqrt{\sum_{j=1}^{m}(s_{i,j} - \overline{s_{i,j}})^2}}. \tag{1}$$

The initial value of a user vector is set to $\boldsymbol{u} = (0, \cdots, 0)$, which means that her preference is unknown. Through interactions between a user and a system, the user vector's value is updated.

We consider the following two types of triggers to update the user vector:

- *question*: a user is asked whether she is interested in the property specified.
- *recommendation*: she is presented with a recommendation and is asked whether she likes it.

In the former case, the value of u_j is directly set from the user's answer to the question. Suppose that the possible answers are YES, NO, or NOT SURE. If the user's response is YES, the value is set to 1, and if her answer is NO, the value is set to -1. Otherwise, the value is set to 0.

In the latter case, the value of u_j is updated based on her feedback in response to the recommendation. We categorize the possible feedback from the user as LIKE, DISLIKE, or NOT SURE. If her feedback is LIKE, the property value of the user vector is increased if the same property of the recommended item is positive, and it is decreased if the same property of the recommended item is negative. If her feedback is DISLIKE, the property value of the user vector is decreased if the same property of the recommended item is positive; it is increased if the same property of the recommended item is negative. Otherwise, the user vector is not changed. More specifically, for recommended item s_i, the user vector is updated as follows:

$$u_j \leftarrow \begin{cases} u_j + s_{i,j} \times \beta & \text{(when user's feedback is LIKE)} \\ u_j - s_{i,j} \times \beta & \text{(when user's feedback is DISLIKE)} \\ u_j & \text{(when user's feedback is NOT SURE).} \end{cases} \tag{2}$$

Here β determines how much a user's feedback to a recommendation affects her preferences.

In the above model, even though we set the number of answer choices to three to simplify the user interface, increasing the number is easy, as in a typical 5-star rating system, by adjusting the value of β.

The overall processing flow is triggered by a user who inputs a certain text such as Start (Fig. 1). First, the user vector is initialized to $(0, 0, \cdots, 0)$. Then the similarity between an item and a user is calculated by formula (1). When an item's similarity exceeds the *recommendation threshold* (α), the item with the highest similarity is presented as a recommendation. Here the *recommendation threshold* is adjusted as described in the next section. The feedback from the user to the recommended item is reflected on the user vector by formula (2).

3.2 Conversation Strategy

In the above conversation control flow, the *recommendation threshold* (α) determines whether a recommendation is made. A bigger value of α 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 questions and recommendations by properly setting the value of α.

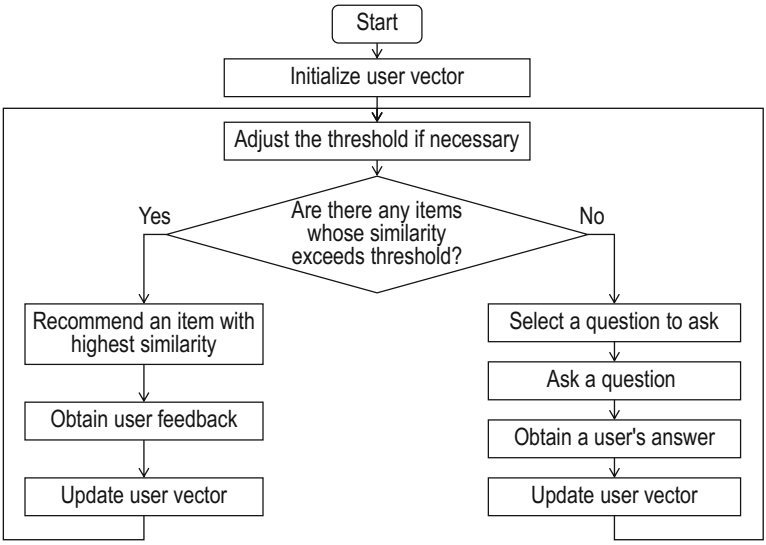


Fig. 1. Overall processing flow

As one heuristic, we decrease the *recommendation threshold* (α) by multiplying by the *threshold adjustment* γ ($\gamma \leq 1$) to increase the chance of successful recommendations when additional recommendations are requested.

The order in which questions are asked is also important. For example, if all the items that might be recommended have value 0 for a certain property, asking a question about that property may be ineffective. Thus, as another heuristic, we calculate a set of items that might be recommended (more specifically, where its similarity value is not negative), and for each property, we count the number of items in that set whose property value is positive. The property with the highest count number is asked next. In this way, we expect that the number of items will be increased that may have higher similarity.

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 49, and the number of properties is 19. We assigned a value of 2 to the property value of a sightseeing spot if it has a strong tendency about the property, and we assigned a value of 1 if it somewhat has the characteristics of the property. If it lacks such characteristics, the property's value is set to 0.

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.

Table 1. Part of dataset used in prototype

Property	Nature	Resort	Temple	Castle	...	History
Sightseeing spots						
Ohtaru falls	2	1	0	0	...	0
Kochi castle	0	0	0	2	...	1
Chikurin temple	0	0	2	0	...	1
⋮						

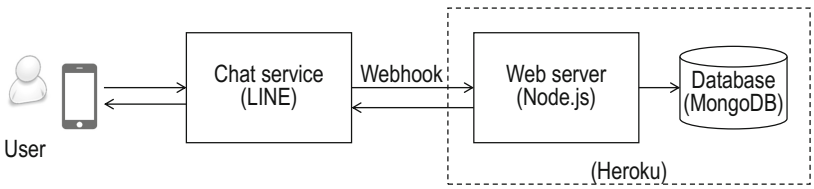


Fig. 2. Overview of prototype system

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 returned to a user. In this prototype, we used Node.js to build a web server for Webhook, which is deployed on the Heroku cloud application platform[2] (Fig. 2).

The LINE messaging API allows us to send not only text messages but also to use GUIs like buttons or links to web sites. In the prototype, we used buttons to let a user input her responses. Figure 3(a) shows an example screenshot that asks a question, and Fig. 3(b) shows the recommendation of an item to a user and a request for feedback about it. When a recommendation is made, its thumbnail picture and a link to a relevant web site are also shown to provide information to determine her feedback on a recommended sightseeing spot.

5 Evaluation Experiments

5.1 Recommendation Threshold

To investigate the effects of *recommendation threshold* (α) that determines the balance between questions and recommendations, we conducted a simulation

[1] https://line.me/en/.
[2] https://www.heroku.com/.

(a) Question (b) Recommendation

Fig. 3. Chatbot user interface

with different user models using the above dataset. With different α values, we counted the number of questions before the first recommendation is made or the system fails to find a suitable recommendation. Since the system has 19 properties, the maximum number of questions is 19.

Figure 4 shows the average, maximum, and minimum counts with eight different user models, which specify the properties that might interest users. Here we presume eight types of typical tourists who visit Kochi prefecture and created a user model by manually defining a user vector for each type. Since a suitable recommendation varies for each user model, the difference between the maximum and minimum counts is rather large. However, the overall result indicates that a higher α generally leads to more questions before the first recommendation is made. Thus, by properly setting the α value, we expect to forge a better balance between questions and recommendations.

5.2 User Study

We also conducted preliminary evaluation experiments with human users. Based on the above simulation results, we set the value of *recommendation threshold* (α) to 0.6. Other parameter values were determined as follows after pretrial experiments: the *threshold adjustment* that determines the amount of decrease of the recommendation threshold value was set to 0.9 ($\gamma = 0.9$) and the feedback coefficient that determines the amount of changes in the user vector after feedback to a recommendation is set to 0.2 ($\beta = 0.2$).

We asked 12 university students in their twenties who also regularly use the LINE service to try our prototype deployed as a LINE bot and to answer questionnaires afterwards. They registered our prototype as a *friend* in their LINE service for their smartphones. The questionnaire consisted of five questions with four choices: strongly agree, agree, disagree, and strongly disagree. The questionnaire also

included a free description text format that sought general comments about the prototype system.

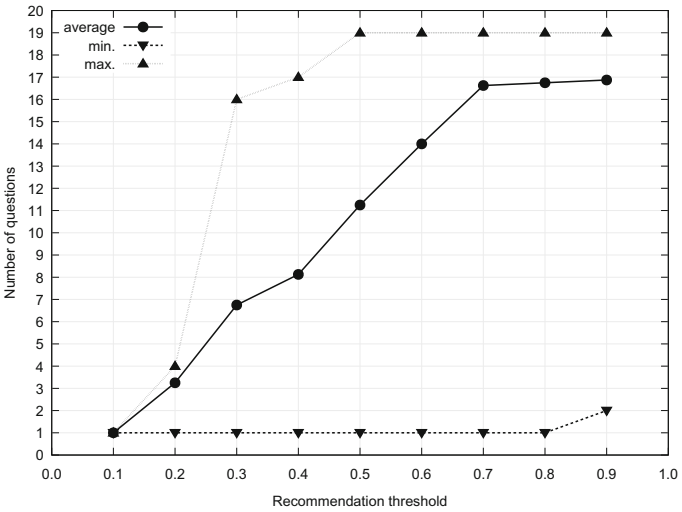
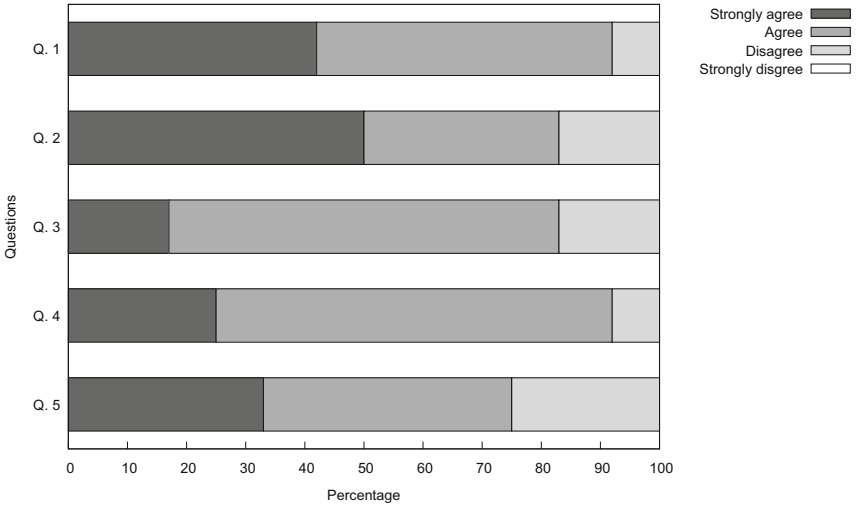


Fig. 4. Number of questions asked before first recommendation



Q. 1: Is the order of questions natural?
Q. 2: Are the questions and recommendations naturally combined?
Q. 3: Is the feedback about recommended items properly considered?
Q. 4: Is the system easy to use?
Q. 5: Do you want to use the system for a different region?

Fig. 5. Questionnaire results

The results of the multiple-choice type questions are shown in Fig. 5. The overall response was positive; combining the questions and recommendations was especially well received by most users (Q. 2). This indicates that the proposed conversation strategy worked well with this domain. Compared with this result, handling feedback to recommendations has room for improvement (Q. 3). The free descriptive question results mostly addressed the viewpoint of a practical system; for example, one comment suggested that not only sightseeing spots but also restaurants in the region should be included in the recommendations.

6 Conclusion and Future Work

This paper described a conversation strategy that intersperses recommendations with questions for user preferences. In addition to directly asking about user preferences, we also considered feedback from users about the recommendations they received. Our simulation results indicate that by changing a parameter value, a balance can be achieved between the number of questions and recommendations. In addition, we implemented a prototype chatbot on the LINE messaging service whose preliminary user evaluation results indicate that it was well received by users.

In the current recommendation model, we did not consider the relationships between properties. In some situations related questions should be asked together. Exploiting the relationships among the properties is one future issue.

In our user evaluation experiments, we defined parameter values based on pretrial results. Since appropriate parameter values depend on the system's dataset, how to determine them is another future issue. When parameter values are determined, subjective impressions about the system must be addressed.

The domain of the current prototype is the recommendations of sightseeing spots in a particular region. Since our proposed approach is applicable to other domains, we plan to expand it to investigate its effectiveness.

Acknowledgment. This work was partially supported by JSPS KAKENHI Grant Number 15K00324.

References

1. Arima, S., Kuroiwa, S., Horiuchi, Y., Furukawa, D.: Question-asking strategy for people with aphasia to remember food names. J. Technol. Persons Disabil. **3**, 10–19 (2015)
2. Augello, A., Scriminaci, M., Gaglio, S., Pilato, G.: A modular framework for versatile conversational agent building. In: 2011 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 577–582. IEEE (2011)
3. Chen, L., Pu, P.: Critiquing-based recommenders: survey and emerging trends. User Model. User-Adapt. Interact. **22**(1), 125–150 (2012)
4. Christakopoulou, K., Radlinski, F., Hofmann, K.: Towards conversational recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 815–824. ACM, New York (2016)

5. Kuwabara, K., Iwamae, T., Wada, Y., Huang, H.-H., Takenaka, K.: Toward a conversation partner agent for people with aphasia: assisting word retrieval. In: Czarnowski, I., Caballero, A.M., Howlett, R.J., Jain, L.C. (eds.) Intelligent Decision Technologies 2016. SIST, vol. 56, pp. 203–213. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39630-9_17
6. Kveton, B., Berkovsky, S.: Minimal interaction search in recommender systems. In: Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI 2015), pp. 236–246. ACM (2015)
7. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, HT 2009, pp. 73–82. ACM, New York (2009)
8. Ricci, F., Rokach, L., Shapira, B. (eds.): Recommender Systems Handbook. Springer, US (2015). https://doi.org/10.1007/978-0-387-85820-3
9. Shi, Y., Larson, M., Hanjalic, A.: Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput. Surv. **47**(1), 3:1–3:45 (2014)
10. Smyth, B., McGinty, L.: An analysis of feedback strategies in conversational recommenders. In: The 14th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2003), pp. 211–216 (2003)
11. Widyantoro, D.H., Baizal, Z.: A framework of conversational recommender system based on user functional requirements. In: 2014 2nd International Conference on Information and Communication Technology (ICoICT), pp. 160–165. IEEE (2014)
12. Yan, M., Castro, P., Cheng, P., Ishakian, V.: Building a chatbot with serverless computing. In: Proceedings of the 1st International Workshop on Mashups of Things and APIs, MOTA 2016, pp. 5:1–5:4. ACM, New York (2016)
13. Zhao, X., Zhang, W., Wang, J.: Interactive collaborative filtering. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM 2013, pp. 1411–1420. ACM, New York (2013)