ReDBot: Exploring Conversational Recommendation for Decision-Making Support in Group Chats

Qiushi Han
Sun Yat-sen University
Zhuhai, China
hanqsh@mail2.sysu.edu.cn

Haoxiang Fan
Sun Yat-sen University
Zhuhai, China
fanhx6@mail2.sysu.edu.cn

ABSTRACT

Group Decision-Making (GDM) commonly takes place online, e.g., in text-based group chats, for daily tasks like choosing a movie or a restaurant. However, reaching a consensus among members in GDM tasks online is non-trivial due to the high workload of collecting necessary information and low awareness of group preferences. In this paper, we explore the design and impact of conversational recommendation for GDM support. Inspired by theories of GDM, we propose a ReDBot that asks questions to identify the group preference and recommends alternatives that match the group preference. We power ReDBot with recent large language models to handle the conversational flow. Our preliminary user study with four three-member groups suggests that ReDBot could reduce members’ workload in collecting information, improve awareness of group preferences, and boost consensus-reaching in GDM group chats. We conclude with design considerations for GDM support.

CCS CONCEPTS

• Human-centered computing → Collaborative and social computing.

KEYWORDS

Group decision-making support, chatbot, large language models, conversational recommendation

ACM Reference Format:


1 INTRODUCTION AND RELATED WORK

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHCHI 2023, November 13–16, 2023, Denpasar, Bali, Indonesia
© 2023 Copyright held by the owner/authors(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1645-4/23/11...$15.00
https://doi.org/10.1145/3629606.3629615

Text-based group chats in instant messaging (IM) apps, e.g., WeChat, Slack, and Skype, offer a convenient communication channel for daily group decision-making (GDM) tasks [10, 26, 30]. For example, a research team or a small group of classmates can chat in IM apps to determine a restaurant, a travel plan, or a movie for their group activities. Taking the task of choosing a restaurant as an example, the group dynamic theory indicates two key steps in a typical GDM process [7, 14]. First, the group members need to collect and exchange information (e.g., dishes, price [12, 23]) related to the decision-making task [14]. Traditionally, members can search their interested information independently and share it in group chats [9, 24], which could be time-consuming due to the extra effort on switching among different interfaces and coordination of the GDM process [4, 25]. Existing works have explored GDM support tools by integrating collaborative search components in group chats [8, 16, 31]. For example, Capra et al. proposed ResultsSpace that supports small groups of users in conducting asynchronous collaborative searches and showed that the collaborative search tools with communication features can support better collaboration [6]. However, the collaborative search process in these works is often unstructured, which may lead to a chaotic discussion in the text-based group chats [5, 15, 21].

Second, the group members need to actively discuss the collected information and their preferences on the alternatives to reach a group consensus [14]. Nevertheless, members in the group chats often lack awareness of each other’s efforts and preferences in this process [6, 18], which could reduce the efficiency of GDM. Previous research has explored the visualization designs to promote awareness of group preferences [1, 17, 18]. For example, Hong et al. developed Collaborative Dynamic Queries (C-DQ) that enables a group to filter queries with visually externalized group awareness and demonstrated its effectiveness for reducing communication costs in the restaurant selection tasks [18]. Nevertheless, little work explores a naturally conversational way to help members discuss their preferences.

In this paper, we explore a chatbot moderator embedded in existing IM apps to iteratively support the two key steps of GDM process in text-based group chats. Our focus is motivated, on one hand, by the potential of a chatbot moderator to naturally structure the information collection step via conversational recommendation, which usually asks questions to elicit preferences and generates suggestions via multi-turn conversations [19]. Existing work
Figure 1: Screenshots of ReDBot’s conversational recommendation process in a group chat for restaurant selection. All messages are translated from Chinese. (a) Self-introduction. (b) Guidance on structuring the chat and questions for eliciting preference. (c) Individual preferences. (d) Recommended restaurants.
ReDBot: Exploring Conversational Recommendation for Decision-Making Support in Group Chats

Figure 2: Workflow of ReDBot in supporting group decision-making (GDM) tasks. Note: `<individual-preferences>` (Figure 3b), `<group-preferences>` (Figure 3a), and `<RECOM>` (Figure 3c) are three kinds of blocks.

(e.g., 3-5 minutes). Members can send “@ReDBot We have completed discussion” to inform ReDBot that they finish the discussion on the current question. ReDBot then prompts GPT-4 with the conversation records to extract individual preferences and check their intersection to examine if the group reaches an consensus (Figure 2b).

Role 2. ReDBot provides information about the alternatives that match individual and group preferences, and helps a group to identify if there is a feasible alternative that could lead to a decision. If the group reaches / does not reaches a consensus on the current question, ReDBot will display the group / individual preferences and proceed to the next question (Figure 2c1, c2). Members can also query the preferences with “@ReDBot Show individual / group preferences” whenever needed. Upon completion of the four questions, ReDBot will generate a sort list of recommended restaurants based on the group preferences. Following the design of popular restaurant apps (e.g., Meituan) in China, it structures the name, rating, number of reviews, cuisine, price, matching degree, and a photo of each restaurant in a preview block (Figure 3c). Members can click “Details” to check the business time, a summary of user reviews, and the URL of the restaurant in a pop-up block. In the same pop-up block, they can also score their likeness of the restaurant to inform others of their preferences.

2.1 Implementation

Restaurant dataset. We use a subset of Yelp Dataset 1 to support the restaurant selection task. As described in the user study below, we simulate the task in the Philadelphia area. Specifically, we randomly select 1146 restaurants with less than 100 reviews in this area. This choice allows us to evaluate ReDBot with an affordable price to invoke GPT-3.5 for processing the reviews. Specifically, to provide an overview of all reviews of each restaurant, we summarize the key aspects of reviews by invoking GPT-3.5 with the following prompt: The following text quoted by three backticks contains multiple reviews in the format of ‘review id: review text’. Summarize each review into some short labels. Then only print out ‘review id: list of labels’ in the response as a JSON file.”[reviews]”. We tag each restaurant with cuisine labels in Chinese based on the categories attribute in Yelp and add its URL and level (1-4) of price 2 via Yelp API. After processing, we have name, address, business hour, rating, number of reviews, URL, price, cuisine, reviews indices, and photos indices for each restaurant.

Group preference. We first prompt GPT-4 to extract individual preferences from the conversation records. The key snippet of the prompt is: “[…] You need to elicit the individual preferences of a group based on the following requirements. There are multiple people who want to choose a restaurant for dinner. They will only consider the ‘cuisines, price, meal time, and ratings’ of restaurants. I will send you their conversation records in the form of: ‘A: [message]. B: [message]. C: [message] ...’. You need to help me elicit the preferences of each member. Please reply in the following strict format: ‘A: [description of the preferences]; B: […] ; C […]’. Notice: [see more requirements on the processing in Appendix A].” Then, if the group reaches a consensus, ReDBot takes the intersection of extracted individual preferences as the group preferences. For example, if member A prefers the restaurant with the 4.2-5 rating score interval, and member B prefers 4-4.8, then the group preference for the rating is 4.2-4.8. On the contrary, if there are still some conflicts, ReDBot would simply record individual preferences instead of calculating group preferences as they are only used for presentation and not for later recommendation generation.

Recommendation. Following the design goals of group recommendation [13], individual preferences here are considered equally. Taking individual preferences as input, ReDBot first uses the group-inclusive method [18] to filter restaurants that satisfy at least one

1https://www.yelp.com/dataset

2Yelp dataset does not have price information, but the Yelp API offers levels of price. Higher level indicates a more expensive price.
Figure 3: Design of blocks sent by ReDBot in supporting the restaurant selection task. All content is translated from Chinese. (a) Group preferences. (b) Individual preferences. (c) Preview of a recommended restaurant. (d) Scoring function for indicating their likeness of the restaurant. (e) Detailed information on the recommended restaurant.

Specifically, ReDBot obtains group preferences by taking the union of individual preferences and filters restaurants according to them. Then, it ranks the filtered restaurants based on majority voting \(I\) (i.e., the more individual preferences a restaurant matches, the higher rank it would get), ratings, and the number of reviews. In other words, regardless of the group’s consensus, top-ranked restaurants will satisfy more individual preferences, have higher ratings and more reviews. Following [13], ReDBot offers a constraint-based explanation by displaying the matching degree between the restaurant and the individual preferences.

3 PRELIMINARY USER STUDY

To evaluate the user experience of ReDBot and inform future design of conversational agents for group decision-making (GDM) support, we conduct a preliminary user study with 12 participants (seven females, age: \(M=22.17, SD=2.59\)). We randomly assign them to four groups (G1-4), each with three members (P1-3, 4-6, 7-9, 10-12). Eight participants report having experiences in GDM tasks in group chats for more than five times, and nine users feel it is difficult to reach a consensus on GDM tasks. All participants are university students in China and are proficient in reading in English.

Task. We simulate a restaurant selection task following previous GDM research [18, 32]. To probe the impact of members’ relationships on user experience with ReDBot, we provide a background story 1 for G1-2 in which members should be familiar with each other, and a story 2 for G3-4 in which members should not. The stories are: “You and two colleagues happen to be on a business trip for different affairs in Philadelphia, USA. (Story 1) You know each other very well. / (Story 2) You are not familiar with each other. You are in a good mood and decide to have dinner together. Prior to the offline meeting, you are going to find a restaurant that is generally acceptable to all of you in the Slack group chat. You are going to use the assistant ReDBot in the chat for the restaurant selection task.”
You do not need to worry about the location of the restaurants as the travel expenses can be reimbursed.

Procedure. We conduct the study remotely. Before the experiment, participants are asked to read documents about task instructions, descriptions of common cuisines, and the introduction of ReDBot. At the appointed experiment time of each group, members log in to Slack with given accounts and join the assigned channel for the group chat. Next, the experimenter in the same channel confirms if all members have read and understood the task materials. Members then start the GDM task with ReDBot, which will ask four questions to elicit their preferences, recommend restaurants, and encourage discussions as described in section 2. Members can inform the experimenter in the channel of the final decision. After that, they are asked to fill in a post-survey about ReDBot’s user experience.

Measure. In the post-survey, we measure participants’ perceptions with the conversational recommendation and GDM support [18, 19, 22] from ReDBot. For the conversational recommendation, we have five questions in 5-points Likert Scale with “5 - Strongly agree”. They are: (acc_pre - accurate preference extraction) “ReDBot accurately extracted and aggregated the group preferences”; (suit_rec - suitable recommendation) “ReDBot provides suitable recommendations that match the group preferences”; (rec_fairness, rec_consensus, and rec_optimality) “The group members reach a high level of consensus on the final decision”; (satisfaction) “I am satisfied with the final group decision”. Apart from the Likert-scale questions, we also ask participants three open-ended questions about ReDBot’s conversational workflow, pros and cons, and suggestions for improvement.

4 RESULTS

We calculate the means for the results of the scale questions, conduct thematic analysis on the answers to the open-ended questions. Figure 4 summarizes the user perceptions of the conversational recommendation and GDM support from ReDBot.

Conversational recommendation. In general, participants give moderate scores to the perceived accuracy of ReDBot’s preference extraction (M = 3.75, SD = 0.97) and the perceived suitability of the recommended restaurants (M = 4.00, SD = 0.60). “I think it did well in capturing our preferences and selecting suitable restaurants for us” (P9). These results suggest that the large language models like GPT-4 had acceptable zero-shot performance in extracting individual preferences from the conversation history in the group chat, but there is a room for improvement, e.g., by fine-tuning it with labeled data. Nevertheless, participants provide relatively high ratings on the measures of rec_fairness (M = 4.17, SD = 0.58), rec_consensus (M = 4.25, SD = 0.62), and rec_optimality (M = 4.33, SD = 0.65). These ratings indicate that ReDBot did well in explaining how individual preferences contribute to the recommended restaurant [13]. P3 expresses his liking of the conversational recommendation feature in response to the open-ended question: “ReDBot offers a more interesting and effective experience for the restaurant selection task in group chats. I can directly interact with it in the same conversational channel rather than switching to another restaurant app as did in the traditional way”.

Group decision-making (GDM) support. Participants generally agree that ReDBot helped them be aware of other members’ preferences on the restaurants (M = 3.92, SD = 0.79). They also find ReDBot helpful in easing the understanding of the group preferences (M = 4, SD = 0.74). “It is useful and intuitive to report the matching degree between the recommended restaurant and the group preferences. I can check others’ preferences, which could make the discussion more efficient” (P11). As for ReDBot’s impact on the GDM process, participants appreciate its features that could make it easier (M = 4.25, SD = 0.45), faster (M = 4.0, SD = 0.43), and more comfortable (M = 4.25, SD = 0.45) to reach the final decision. Four participants (P1, P7, P11, P12) highlight the value of displayed group and individual preferences, matching degrees, and the summary of reviews in facilitating GDM. “I like the automatic aggregated preferences after each question. They are intuitive and helpful for group decision-making. The summarized reviews provide keywords of each restaurant, which make it convenient for the users to collect needed information” (P11). P11 further comments: “ReDBot would make the GDM process more comfortable for people who have difficulties in choosing what to eat”. Besides, participants perceive that ReDBot could improve the fairness (M = 4.0, SD = 0.6) of and members’ confidence (M = 4.08, SD = 0.51) on the final group decision. They feel that ReDBot promotes the perceived consensus level among group members (M = 4.42, SD = 0.51) and improve their satisfaction with the final decision (M = 4.58, SD = 0.51).

Relationship among group members. Our preliminary user study also indicates the potential impact of the relationship among group members on the measured user experience. As described in the user task (section 3), members in G1-2 should act like they are familiar with each other, while those in G3-4 should act like they are not. Specifically, in the 12 out of the 14 Likert-scale measures (Figure 4), members in G3-4 generally give higher scores compared to those in G1-2. For example, members in G3-4 feel that ReDBot performs quite well in providing fair recommendations (M = 4.33, SD = 0.52), and the whole group is likely to agree with the recommendations (M = 4.67, SD = 0.52). However, members in G1-2 generally give lower ratings on these two measures of fair_decision (M = 3.67, SD = 0.52) and consensus (M = 4.17, SD = 0.41). These preliminary results can inform HCI and CSCW researchers to systematically explore the impact of members’ relationships on the effectiveness and user experience of group decision-making support tools.

5 DISCUSSION
In this work, we propose ReDBot that supports the information collection and preference discussion steps in the group decision-making (GDM) process in group chats. For the formal step, results of our preliminary user study suggest that ReDBot’s conversational recommendation could save group members’ effort in searching information. We extend the usage of conversational recommendation from individual to multi-party scenarios. Our workflow (Figure 2) and message design (Figure 3) of ReDBot can serve as a starting point for the design and development of future conversational GDM support tools. For the later step, our user study shows that ReDBot’s extracted, aggregated, and structured user preferences could help group members be aware of each other’s preferences and help them reach a group consensus more easily and comfortably. Our findings complement previous work that demonstrates the usefulness of chatbot moderators in group chats. We also showcase the huge potential of leveraging recent large language models to power the chatbot moderator, e.g., for zero-shot preference extraction from conversation history.

Our work also offers two lessons that lead to the design considerations for future GDM support tools in group chats. First, we observe cases where group members did not follow ReDBot’s guidance to share and discuss their preferences on the current question. This can lead to inaccurate preference extraction and harm the user experience of ReDBot. We therefore suggest that future GDM support tools should have more systematic dialogue management module, e.g., with intent classifiers that track the progress of the conversation. Second, we use blocks (e.g., Figure 1e and Figure 3) to display the information of recommended restaurants in the Slack channel. This could take up a large space of the group chat window and make it inconvenient for group members to track the conversation history. Future GDM support tools should simplify the message and interaction design. For instance, they could fix a panel next to the group chat window and use it for displaying message blocks and supporting the interaction with the tool. This may require development of a new instant messaging app (e.g., the one like [18]) or available development APIs of existing apps.

Our preliminary user study has several limitations that call for future work. First, we collect qualitative feedback on ReDBot’s user experience and plan to conduct a comparative study including behavioral measures to quantitatively explore its effectiveness against the baseline condition without ReDBot. Second, we simulate a GDM scenario in the user study following [18, 28]. It would require a field study to evaluate ReDBot in real-world GDM scenarios. Third, we conduct the user study with four three-member groups to collect first-hand feedback for improving ReDBot. Future work should involve more participants, examine groups of a larger size, and conduct statistical comparisons between different conditions in the formal user study.

6 CONCLUSION

In this paper, we design and develop ReDBot that uses conversational recommendation to support group decision-making in a restaurant selection task. ReDBot leverages the large language models to extract the user preferences from the conversational history in group chats and summarize the reviews of the restaurants. Our preliminary study with 12 participants suggests that ReDBot could facilitate group members to collect necessary information, be aware of group preferences, and make the group decision more comfortably. We hope that our work can serve as a good starting point for future conversational agents to support daily group decision-making tasks.

ACKNOWLEDGMENTS

This work is supported by the Young Scientists Fund of the National Natural Science Foundation of China with Grant No. 62202509.

REFERENCES


problems and incomplete preferences. Information Sciences 180, 23 (2010), 4477–4495.


A TEMPLATE OF THE PROMPT FOR PREFERENCE EXTRACTION

As previously mentioned, ReDBot extracts individual preferences from conversation records by prompting GPT-4. And here, we present more details about the prompt.

A.1 English Version Transcript from Chinese

You are a high-performance preference extractor. You need to elicit the individual preferences of a group based on the following requirements: There are multiple people who want to choose a restaurant for dinner. They will only consider the "[characteristic]" of restaurants. I will send you their conversation records in the form of: A: [message] B: [message] C: [message] … You need to help me elicit the preferences of each member. Please reply in the following strict format: A: [description of the preferences] B: [...] C: [...] 

Notice: 1. [Requirements on specific dimensions]. 2. Before eliciting preferences, you have to go through the conversation records to determine whether someone in the conversation records did not mention his preferences, for example: Said something irrelevant to the current characteristic of the restaurant or said nothing, then think he didn’t mention his preference. 3. If there is someone who has not mentioned his preference, then his personal preference list can be directly output as None (e.g., id: None); otherwise, it is considered that he mentioned his preference, and referred to the following steps to process them. (Note: Messages like "hahaha" that have no actual meaning are also considered as not mentioning their preference) 4. Each person’s preference only outputs the final result once, and each person’s preference corresponds to one line. The reply can only contain the above content, and you must not reply with any redundant content!!!

5. You should process each conversation records from front to back, and then keep updating everyone’s personal preferences.
6. Output after processing all the conversation records, and everyone’s preference is to output only the latest version when outputting, that is, do not output the intermediate preferences!!!

The conversation records are: [records]

A.2 Chinese Version Used in the Study

你是一个性能卓越的偏好提取器。你将按照如下要求提取小组成员的个人偏好：现在有一群人想要选择聚餐的餐厅。他们在选择餐厅时只会考虑餐厅的“特征”。我会把聊天记录以：A: [消息] B: [消息] C: [消息] ... 的形式发给你; 你需要帮我提取他们每个人的偏好。请严格按照如下格式回复：A: [偏好描述] B: [偏好描述] C: [...]

注意：
1. [具体维度的要求]。
2. 在提取偏好之前你要先过一遍聊天记录，判断聊天记录中是否有人没提及自己的偏好，例如：说了一些与当前需要考虑的餐厅特征不相关的东西或啥也没说，纷纷表示他没有提及自己的偏好。
3. 如果还有人没有提及自己的偏好，则他的个人偏好列表直接输出为 None 即可 (id: None)。否则，认为他提及了自己的偏好，并参照下面的步骤处理他的偏好。(注意："哈哈哈" 这种没有实际意义的消息也认为没提及自己的偏好)
4. 每个人的偏好只输出一次最终结果，且每个人的偏好对应一行，回复只可包含上述内容，一定不要回复任何多余的内容！！！！
5. 你应该从前往后处理每条聊天记录，然后不断更新每个人的个人偏好。
6. 处理完全部聊天记录后再输出，且输出时每个人的偏好只输出最新的版本，即不要输出中间的过程!!!!!!!!!!!! 聊天记录：[群聊记录]

Figure 6: Chinese example.

A.4 Example Output

- A: [4.5, 5.0]
- B: [0, 4.8]
- C: [4.0, 5.0]
- D: [4.5, 4.8]

Figure 5: Chinese prompt.