

Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Ellie S. Paek

March 29, 2024

Emora Assistant Bot: Revolutionizing Task Automation and Chatbot Evaluation

By

Ellie S. Paek

Jinho D. Choi

Advisor

Department of Computer Science

Jinho D. Choi

Advisor

Bethany Mamola
Committee Member

Talea Mayo
Committee Member

2024

Emora Assistant Bot: Revolutionizing Task Automation and Chatbot Evaluation

By

Ellie S. Paek

Jinho D. Choi
Advisor

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Computer Science

2024

Abstract

Emora Assistant Bot: Revolutionizing Task Automation and Chatbot Evaluation
By Ellie S. Paek

The Emora Assistant Bot, part of the Emora Chat: College Companion project, was developed by members of the Emory NLP Research Lab. Motivated by the call for efficiency in executing tasks, Emora excels in managing both general administrative tasks and those specific to classroom settings, all while facilitating seamless communication among multiple users. Leveraging the Emora State Transition Dialogue Manager framework and OpenAI's GPT-3.5 Turbo API, the Assistant Bot executes seventeen different tasks through natural language interaction, offering users a conversational and efficient experience. An innovative automated evaluation approach utilizing the GPT-3.5 language model is used to evaluate this task-oriented chatbot, providing valuable insights into Emora's performance and highlights areas for improvement. Conducting automatic evaluations revealed limitations with the STDM framework, yet Emora demonstrated successes in information extraction and task categorization, underscoring her capability for seamless task execution. Moreover, despite occasional inconsistencies, the GPT simulation emerged as a promising method for evaluating task-oriented chatbots. Between two professor and 20 student profiles, Emora had an average success rate of 94.3% for task execution, and 94% for natural language understanding. The GPT simulation displayed an average success rate of about 81%. Through this research, the Emora Assistant Bot project emerges as a pioneering solution for automating administrative tasks, showcasing the potential of large language models in both task execution and evaluation within the realm of chatbots.

Emora Assistant Bot: Revolutionizing Task Automation and Chatbot Evaluation

By

Ellie S. Paek

Jinho D. Choi
Advisor

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Computer Science

2024

Acknowledgments

I'd like to acknowledge and thank the following groups and people:

Dr. Jinho D. Choi, for his advisory and faith in me to carry out this project. His enthusiasm and support made me enjoy my work in the NLP Lab and inspired me to continue developing Emora to the version she is today.

The current and former members of the Emory NLP Research Lab — James Finch, Sarah Finch, Talyn Fan, Camila F. Calvino Pintado, Susanna Spooner, Leah Smith, and Helen Jin — for their work in Emora College Companion Project. Thanks to their contributions and research, this thesis was made possible.

My friends, suitemates, and instructors for cheering me on through this process. The patience and encouragement that they gave me during my low points is an invaluable gift that I will never forget.

My parents and brother — Moonjung, Hoon, and Jay — who continue to support my actions and my goals to be better and work harder every day. Even though we're often miles apart, I'm always empowered by the love that you send my way.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Merits	4
1.3	Thesis Statement	4
2	Background	6
2.1	Task-Oriented Chatbots	6
2.1.1	One-on-One Chatbots	6
2.1.2	Multiple User Chatbots	7
2.2	Automated Evaluations	9
2.2.1	GPT Evaluations in NLP Tasks	9
2.2.2	Automated Chatbot Evaluations	9
2.3	Preliminary Work	11
2.3.1	Emora Course Assistant and Previous Assistant Bot	11
2.3.2	Dialogue Evaluation with GPT	12
3	Approach	14
3.1	Framework	14
3.2	Database	17
3.3	Features	18
3.3.1	Names and Hub	18

3.3.2	Appointments	18
3.3.3	Messages	21
3.3.4	Groups	21
4	Experiments	31
4.1	GPT Evaluation	31
4.1.1	Evaluation Framework	32
4.1.2	User Profiles	32
4.1.3	Simulation Process	34
4.2	Evaluation Criterion and Metrics	36
4.2.1	Emora Assistant Chatbot	36
4.2.2	GPT Simulation	38
4.3	Results and Discussion	39
4.3.1	Results from Version 1	40
4.3.2	Results from Version 2	43
5	Analysis	47
5.1	Analysis of Emora Errors	47
5.1.1	Inappropriate Responses	48
5.1.2	Improper Input Extraction	49
5.1.3	STDM Limitations	51
5.2	Analysis of GPT Simulation Errors	56
5.3	Possible Explanations and Recommendations	61
6	Conclusion	62
6.1	Future Directions	63
6.2	Research Insights	64

A Appendix	66
A.1 GPT Function Used for Emora’s Input Extraction	66
A.2 GPT Prompts Used for GPT Simulation	68
Bibliography	86

List of Figures

3.1	A diagram of a sample state machine-based conversation between a chatbot and user. The blue dialogue denote Emora’s responses, while the green dialogue represents the key words that the groups of expressions that NATEX may capture from user responses to move to a different conversation state, represented as circles.	15
3.2	An example of a conversation between a user and Emora, where the user’s input is analyzed and the necessary information is extracted. Between the user and system transition, what is extracted from the user’s input using GPT is outlined.	16
3.3	An example of an initial conversation between Emora and a mock user with the name calling function. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	19
3.4	An example of Emora handling an appointment creation request. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	20
3.5	An example of Emora handling a message sending request. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	22

3.6	An example of Emora handling an inbox checking. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	23
3.7	A screenshot of the Emora application, detailing a conversation for sending a message.	24
3.8	An example of Emora handling a group creation request. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	26
3.9	An example of Emora handling a group identification request. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	27
3.10	An example of Emora sending feedback requests to groups. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	28
3.11	An example of Emora recording feedback from group members. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	29
3.12	An example of Emora handling feedback checks and providing a cohesive summary. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.	30
5.1	An example of Emora ignoring the User. The inappropriate response is in red.	48
5.2	An example of Emora responding beyond the conversation context. The inappropriate response is in red.	48
5.3	An example of Emora incorrectly identifying a messaging task within a request. The inappropriate response is in red.	49

5.4	An example of Emora incorrectly identifying a messaging task within a statement of gratitude. The inappropriate response is in red.	50
5.5	An example of Emora incorrectly extracting information from a user's input. The inappropriate response is in red.	50
5.6	An example of Emora not handling the check inboxes tasks due to the STD M State Lock (SL). The inappropriate response is in red.	52
5.7	An example of Emora entering a component and being unable to handle the user's requested task due to STD M State Lock (SL). The inappropriate response is in red.	52
5.8	An example of Emora not handling the user's requested task due to STD M NATEX Issues (NI). The inappropriate response is in red.	53
5.9	An example of Emora not handling the user's requested task due to STD M NATEX Issues (NI). The inappropriate response is in red.	53
5.10	An example of Emora unable to respond appropriately to the user due to STD M NATEX Issues (NI). The inappropriate response is in red.	54
5.11	An example of Emora returning to the name calling component due to STD M Special Character errors (SC). The inappropriate response is in red.	55
5.12	An example of Emora unable to address multiple tasks due to STD M Single Transition issues (ST). The inappropriate response is in red.	56
5.13	An example of the user ignoring the chatbot's last response by responding with a request (IC). The inappropriate response is in red.	57
5.14	An example of the user ignoring the chatbot's last response by repeating themselves (IC). The inappropriate response is in red.	57
5.15	An example of the user repeating themselves (RP). The inappropriate response is in red.	58

5.16	An example of the user not adhering to the provided user information (UI). The inappropriate response is in red. The relevant part of the user profile is provided below the conversation.	58
5.17	An example of the user forgetting the previous conversation history (CH). The inappropriate response is in red.	59
5.18	An example of the simulation generating responses fit for the chatbot rather than the user (RR). The inappropriate response is in red. . . .	60
5.19	An example of the simulation generating responses not addressing Emora, and rather the sender of the message (CP). The inappropriate response is in red.	60

List of Tables

4.1	Rates of Success by Emora and GPT Based on 1 Professor, 10 Student Profiles (Simulation Version 1). The bottom row displays the average of the column's values.	40
4.2	Rates of Success by Emora and GPT Based on 1 Professor, 20 Student Profiles (Simulation Version 1). The bottom row displays the average of the column's values.	41
4.3	Rates of Success by Emora and GPT Based on 1 Professor, 10 Student Profiles (Simulation Version 2) The bottom row displays the average of the column's values.	43
4.4	Rates of Success by Emora and GPT Based on 1 Professor, 20 Student Profiles (Simulation Version 2) The bottom row displays the average of the column's values.	44
4.5	Rates of Success by Emora and GPT Based on 2 Professors, 20 Student Profiles (Simulation Version 2) The bottom row displays the average of the column's values.	45
5.1	Results of the STDM-based error analysis on the Emora Assistant Bot.	51
5.2	Results of the error analysis on the GPT simulation.	56

Chapter 1

Introduction

This thesis introduces the Emora Assistant Bot (Assistant Bot or Emora), one of the three chatbots of the Emora Chat: College Companion project. This chatbot is developed to handle administrative and classroom-based tasks, simultaneously communicating between multiple people to do so. Upon requests for different tasks, Emora extracts the necessary information to carry out these tasks before executing them. She does so while communicating with the user in natural language, allowing for a more conversational experience of requesting a secretary for assistance rather than a menu-based automated experience.

In tandem with Emora, this thesis introduces an automated method to evaluate task-oriented chatbots. Utilizing an asynchronous dialogue framework coupled with GPT, a simulation of multiple users in a classroom setting is built and subsequently used to evaluate the Assistant Bot's capabilities in task execution and understanding of natural language. This evaluation method helps expose the limitations of the Emora Assistant Bot and the next steps in further enhancement for optimal user experience.

The Assistant Bot is part of a collaborative project involving undergraduate and graduate students at the Natural Language Processing (NLP) Lab at Emory University. For this branch of the project, the research questions to explore are as follows:

1. Can general and classroom-oriented administrative tasks for multiple people be successfully automated?
2. Do large language models achieve state-of-the-art user experience in task categorization and information extraction?
3. Can large language models replace humans in evaluating task-oriented chatbots?

The Emora Assistant Bot is developed with the Emora State Transition Dialogue Manager (Emora STD M) framework and OpenAI's GPT API (GPT), allowing for seamless natural language understanding and information extraction, while also making sure the responses do not stray from the designated tasks. Regardless of users' occupation as a student or professor, Emora allows any task to be requested for initiation, giving users opportunities to utilize her for personalized needs. A database is linked to utilizing the Assistant Bot, which stores users' IDs and allows for requests and messages to be transferred across multiple users.

As of the time of writing, the Emora Assistant Bot is accessible through a web-based application and a mobile app on Apple and Android devices. She is able to execute seventeen different administrative tasks through natural conversation, each leading to a different component for necessary information extraction. Each successful execution of a task that involves multiple people pushes a notification to the recipients' application, informing them an asynchronous message or request has been sent to them.

1.1 Motivation

In today's rapidly evolving business and educational landscapes, routine administrative tasks have become increasingly indispensable. The Bureau of Labor Statistics defines these tasks as ones that help maintain efficiency within organizations, including but

not limited to companies and schools [3]. Such tasks include creating appointments with people and delivering messages of important content.

Typically, administrative duties are handled through human involvement. Designated workers such as secretaries and administrative assistants streamlined these organizational processes [3]. However, in increasingly fast-paced environments, the growing demand for efficiency and productivity presents challenges, particularly when tasks must be executed promptly.

The Emora Assistant Bot offers a solution to efficiently clearing these tasks by automating them. With the chatbot's abilities asynchronously deliver messages and appointment requests to users, Emora provides cleaner task management without extraneous content such as advertisements. In addition, managing group-related activities is facilitated through Emora, allowing for needs of different users to be met, whether they require a large group for classroom settings or a smaller group for projects or labs.

One of Emora's notable features is the anonymity system when sending messages and giving feedback, derived from feedback systems that are often implemented in colleges. While schools such as Emory University implement feedback systems for courses at the end of an academic term, there are classes that seek feedback for mid-term activities such as homeworks and exams. However, as these are not implemented by the school, often these feedback forms rely on request from instructors and third-party platforms such as Google or Microsoft Forms. With feedback and messages features with an integrated anonymity system, Emora helps facilitate communication between instructors and students, where instructors are able to easily request feedback. Students, in turn, are able to anonymously give feedback per the instructor's requests or through messages should they wish to.

1.2 Merits

The Emora Assistant Bot introduces several merits, of which its main merit is the ability to communicate and carry out tasks between multiple people at once. With seamless communication between groups of people at once, the bot can execute various tasks that are both administrative and education-based, as well as open to all users rather than restricted to certain roles. This allows for more efficient task management for users, depending on what task they wish to complete.

In addition, the Assistant Bot utilizes GPT as its main framework of natural language understanding (NLU). With GPT's efficient methods of extracting information and categorizing tasks, Emora demonstrates a robust NLU, allowing for task execution even through smooth conversation flow [25, 22].

Finally, this thesis proposes a groundbreaking evaluation approach, introducing a novel method wherein a simulation framework based on GPT is employed to assess the Emora Assistant Bot. By harnessing the capabilities of a large language model, this approach offers superior efficiency and accuracy compared to traditional human evaluations. The adoption of such a method not only streamlines the evaluation process but also holds promise for advancing the optimization of task-oriented chatbots in the future, thereby enhancing the overall user experience.

1.3 Thesis Statement

By developing a chatbot utilizing a state machine-based framework, GPT, and a linked database, administrative tasks will be effectively automated, allowing people to request Emora to handle such tasks, even when involving multiple people at once. By utilizing GPT, requested tasks are effectively categorized and the required information to execute the task is successfully extracted, allowing for smoother transitions and efficient information transfer between the user and chatbot. Finally, by building a

framework that automates Emora's evaluation, this thesis shows that large language models demonstrate strong potential in effectively evaluating task-oriented chatbots.

Chapter 2

Background

2.1 Task-Oriented Chatbots

Utilizing automated methods to execute administrative tasks is not uncommon. Third party tools, particularly calendars such as ones from Google and Microsoft help schedule events and online meetings effectively. However, implementing chatbots to handle some of such tasks is far less common than these tools.

2.1.1 One-on-One Chatbots

Often, chatbots that are implemented to facilitate administrative tasks are one-on-one chatbots: when the user speaks to the chatbot, the chatbot responds in similar or same ways as it does to other users, making the experience among users identical. Examples of this include chatbots that are implemented for handling frequently asked questions (FAQs) [1, 13, 18]. There also exists chatbots that are used for automating interviews, where the bot is able to effectively ask questions and empathically respond to interviewees' responses [23].

These chatbots are typically developed with state or rule-based frameworks, namely Google's DialogFlow which facilitates the development of conversation-based chatbots

with the Google Cloud NLP. However, the emergence of large language models (LLMs) and their versatility in any given NLP task has led to an increased interest of utilizing them as the foundation of such task-oriented chatbots [11, 18].

While many of these FAQ or interview chatbots have been utilized in educational settings, there have been some task-oriented chatbots that have been developed to specifically adhere to the academic audience. Ibna Riza et al. [10] established that task-oriented chatbots may be used for various educational purposes, including acting as an assessment chatbot for students and recommending new material. The presence of online chatbots would help with effective content integration, as well as facilitate engagement and easier access for students [10, 1].

2.1.2 Multiple User Chatbots

Much of such chatbot work established in Section 2.1.1, however, does not extend to multiple people at once. Typically, the interaction is between only the user and the chatbot, as the bot only has access to a database of factual or determined information that it may use to assist the user. Subsequently, most chatbots lack the ability to transfer information between multiple users, as the users' identities are not recorded. Thus, tasks that one-on-one chatbots handled are mostly information-based, and developed to respond to a general user. In this perspective, the Emora Assistant Bot's main novelty is the ability to communicate with multiple users and transfer information between users to assist with various general administrative tasks.

While there haven't been many works aiming to handle multiple users and chatbot conversations, Toxtli et al. [21] developed a chatbot, TaskBot, in 2018 that aimed to communicate with many people at once. TaskBot was integrated into Microsoft Teams, allowing users to interact with the bot within the platform. The chatbot served as a reminder bot, where users request for other users to be reminded to complete specific tasks. While the conversations were one-on-one between the chatbot and the user, the

chatbot is able to transfer information (reminders) between multiple users, facilitating asynchronous communication between them [21].

Unlike the Assistant Bot, TaskBot handles one sole task: to remind others periodically to complete their task. The Emora Assistant Bot is able to handle multiple tasks, and with the notification system, users are able to notify others to complete tasks with the asynchronous messaging system integrated into Emora. In addition, TaskBot is developed using the Microsoft Language Understanding Intelligent Service, allowing for broad natural language understanding (NLU) from the chatbot [21]. However, this NLU was limited, especially when TaskBot is attempting to extract string-based names instead of specific tags that Microsoft Teams offers. The Assistant Bot, by contrast, utilizes a combination of GPT-3.5 and Emora STDm's NATEX as methods of information extraction, which can extract information from long string messages without requiring special characters.

In addition to Toxtli et al. [21]'s TaskBot, Mendoza et al. [17] developed a chatbot designed for interaction between students and teachers. Specifically geared towards middle school, this Web-based chatbot demonstrates specific conversational interactions based on whether the user profile is a student or a teacher — of which it is manually inputted by an administrator. This chatbot, developed with DialogFlow, can carry out multiple functions when requested by the user, which include scheduling events for teachers, sending files between users, and answering FAQs.

With the ability to communicate between multiple users and carry out various administrative and education-based tasks, this chatbot is comparable to the Emora Assistant Bot. However, it is important to note that the chatbot developed by Mendoza et al. [17] is initialized by the administrator, and thus user profiles and designated privileges, such as the ability to schedule events, is locked unless request for a change. The Assistant Bot, by contrast, demonstrates fluidity in the creation of groups, allowing for users to create groups, regardless of whether they are an

instructor looking to make a class, or a student seeking to create a group for a project or studying.

2.2 Automated Evaluations

The concept of automated evaluations is not new to the realm of NLP. Due to the inefficiency of time and resources in evaluations, researchers seek to find more efficient methods to evaluate methodologies, including chatbots.

With the rise of LLMs, in which they have demonstrated state-of-the-art results in numerous NLP tasks, researchers have been attempting to utilize them in evaluations. In particular, OpenAI's GPT has been popular due to its ability to successfully execute multiple NLP tasks, leading it to be used for evaluation tasks [11, 12].

2.2.1 GPT Evaluations in NLP Tasks

The evaluation tasks assigned to GPT has been varied, but more often are dialogue and writing evaluation tasks. GPT-4, the most powerful LLM offered by OpenAI to this date, is used as a baseline to evaluate such tasks, such as natural language generation (NLG) or various human written responses [14, 16]. GPT-4 has also been used to evaluate other LLMs on various sequence-to-sequence tasks [19]. In all of such tasks, it has been reported that GPT-4 often produces high agreement scores with human annotators.

2.2.2 Automated Chatbot Evaluations

While GPT, particularly GPT-4, demonstrates proficiency in evaluating various NLP tasks, its performance in being used for online human evaluation is more limited. There has been research in evaluating chatbots in automated ways, with frameworks developed to facilitate the process. Often, these frameworks utilize the method of

self-talk, where chatbots talk to themselves [5, 9]. These have demonstrated high correlation with human annotated scores, though there are some discrepancies due to the unclear definition of a "natural" response from the bot.

Others have employed the method of making chatbots converse with each other. Deriu et al. [4] and Yang et al. [24] both propose frameworks in which numerous chatbots are contested against one another. The conversations that the chatbots have with each other are evaluated by the humans, and in a contest-like format, each chatbot is scored based on various criterion and subsequently ranked.

One of the exemplary works utilizing GPT for automated chatbot evaluation comes from [20], where they propose the Dialog system Evaluation framework based on Prompting (DEP). DEP utilizes an LLM, namely the InstructGPT model, with a specified prompt to converse with chatbots. The framework demonstrates a high correlation with human evaluation, as well as functionalities in generating varied dialogue and a reasonable approximation of human evaluation.

One important point to note with all of these evaluations, however, is that they are based around evaluating social-based chatbots. While utilizing chatbots and LLMs produce valuable evaluations for chatbots, the standards for evaluating whether a chatbot's response is appropriate to the evaluator's input may be unclear. As the Emora Assistant Bot is a task-oriented chatbot, however, evaluating the functionalities through automated methods will be clearer than that of social chatbots, whether it be on the turn-based or overall level. To that end, the GPT simulator evaluation's main novelty is introducing a method of automated chatbot evaluation on task-oriented chatbots.

2.3 Preliminary Work

2.3.1 Emora Course Assistant and Previous Assistant Bot

Preceding the Emora Assistant Bot, the Emora Course Assistant was a project originally in development in Emory University’s Spring 2023 semester. As a member of a team consisting of undergraduate and graduate students from the Emory NLP Lab, this course assistant chatbot was developed with goals to improve various aspects of the course, while communicating successfully through natural language. The course assistant was developed by only utilizing the Emora STDM framework, as well as linked to database where information was stored. Developers handcrafted components for tasks that the chatbot would handle, including chatbot transitions and responses that would be required to shift to a new conversation state.

The Emora Course Assistant, at the time of writing, was able to handle different tasks depending on the occupation of the user talking to her: student, professor, and teaching assistant. Professors and teaching assistants could adjust their office hour times, while teaching assistants and students can make appointment requests to their respective professors. In addition to appointments, Emora could find students to help other students on certain assignments or studying, as well as form study groups between students. Students were also able to offer anonymous feedback to the professor, as well as request for any accommodations. Any of the components of which its activity could be quantified (for example, two students requesting for other students’ help through Emora) was stored. Upon the request of the professor of the class, Emora could compile statistics to present to them.

During the development of the course assistant, however, several problems began to surface, many of them being related to utilizing the Emora STDM and its NATural language EXpression (NATEX) NLU to develop chatbot transitions. While hand-crafting chatbot responses helped maintain consistency in natural language generation

for Emora, handcrafting combinations for the NLU using NATEX showed limits in extracting the necessary information for carrying out tasks, especially when trying to categorize the requested tasks. Human responses to conversation can be diverse, of which not all may be captured by the chatbot transitions. In addition, since the NATEX for each transition was crafted by a developer on the team, the chatbot’s understanding within those components is limited to what was put in by the developer. Extensive testing and adjustment of the NATEX can possibly resolve the issue, but even then, the scope of human responses can be large and ambiguous, in which trying to capture all of them may lead to ambiguous or perhaps incorrect conversational state transitions.

As a result, much of the chatbot responses had to be adjusted such that the range of possible user responses would be small, such as answering a yes or no question from Emora. However, this resulted in the course assistant becoming less of a conversational chatbot, which went against the scope of this project.

As a result, in the summer, a member of the team developed a new iteration of the course assistant by replacing the NATEX NLU with GPT-3.5 functions. This allowed for the chatbot to more effectively extract information and categorize tasks, resolving many of the issues that formed with the development of the course assistant.

This iteration of the chatbot was the precursor to the current Assistant Chatbot, where the main task she was able to handle was making and confirming appointments. Through the progress of this project, other task-handling components were additionally created on this chatbot, leading to the iteration of the Assistant Chatbot that is introduced in this thesis.

2.3.2 Dialogue Evaluation with GPT

In tandem with the course assistant chatbot work, sets of human-chatbot dialogue from Finch et al. [7] was evaluated with GPT-3.5 in collaboration with a graduate student

of the Emory NLP Lab. After few-shot prompting for the ideal prompt, GPT was used to evaluate sets of human-chatbot dialogue to determine if behavior-detection task evaluation could be automated through LLMs, replacing human evaluation as possibly a more cost-effective option. The results of the GPT evaluation were compared against the human evaluations from Finch et al. [7], as well as other models specialized for particular behavior-detection task.

Through the evaluations, it was discovered the GPT does not have exemplary performance compared to human evaluations, often falling short [8]. Despite this, it did often outperform specialized models for specific behavior labels, despite not being finetuned with a large sample of specific examples. This shows that GPT demonstrates strong potential in dialogue evaluation, especially as a low-cost alternative.

While the research scope in Finch et al. [8] and this thesis may differ, the evaluation of GPT in dialogue showcases its potential as a cost-effective alternative to human evaluation, yielding comparable results. Given its promising capabilities as an external evaluator in human-to-chatbot dialogue, this thesis seeks to explore GPT's potential as an internal evaluator, supplanting human evaluation entirely.

Chapter 3

Approach

3.1 Framework

The foundation of the Emora Assistant Bot uses the Emora State Transition Dialogue Manager (Emora STD M). Developed by Finch and Choi [6], Emora STD M demonstrates a framework with adjustable dialogue states that helps rapidly prototype chatbots. Emora STD M develops state machine-based chatbots, where each state corresponds to the state of the conversation (what kind of conversation the chatbot and the user are having) and each transition corresponds to a response from the system (the chatbot) or the user. Using this framework, each response from the chatbot is handcrafted by the developer, and anticipated responses from users help move the chatbot to a different conversation state. Each different conversation state results in a different response from the bot, ensuring diversity in responses.

To facilitate the chatbot’s NLU, Emora STD M also introduces the NATural language EXpression, or NATEX. This is a method of matching expressions to user input via string and ontology matching [6]. This allows chatbots utilizing this framework to understand user inputs through specific key words and subsequently respond appropriately. Figure 3.1 demonstrates a dialogue flow using STD M, where each transition

(shown as arrows) to a different conversation state (shown as circles) is determined by the user’s response. Using NATEX, the transition is determined through specific key word groups, such as ”make appointments” moving the conversation state to a component where appointments are created.

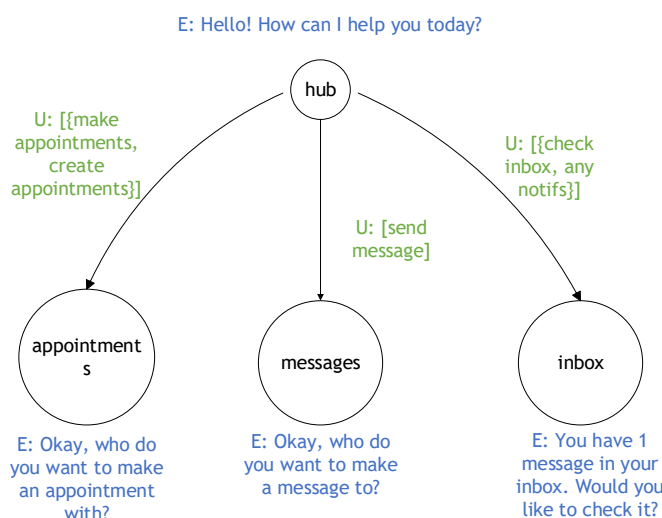


Figure 3.1: A diagram of a sample state machine-based conversation between a chatbot and user. The blue dialogue denote Emora’s responses, while the green dialogue represents the key words that the groups of expressions that NATEX may capture from user responses to move to a different conversation state, represented as circles.

While NATEX does assist with expanding the NLU for a prototype chatbot, there is a limit in which the chatbot may be able to understand for an expanded range of tasks that it is requested to do. Since it consists of manual vocabulary and sentence matching via the developer, the chatbot’s understanding is limited to the developer. In addition, since an administrative assistant’s tasks involve extracting specific yet various information from the user — including what tasks that need to be addressed, or what times they may be available — generating NATEX to find every possibility of the information extraction may be inaccurate and inefficient (see Section 2.3.1).

Thus, many of the NATEX transitions are replaced with functions that utilize the GPT-3.5 API, which has demonstrated potential in excelling in NLU with few-shot

prompting [15, 2, 25, 22]. Specifically, GPT-3.5 is used for categorizing tasks requested by the user, as well as extracting information required to complete said tasks. Each function with GPT is integrated into the STD M’s dialogue flow using macros, which are the Emora STD M’s built-in functions that can be used within conversation states and allow external Python functions to be executed with their usage.

In each user transition, depending on which conversation state Emora is in, a GPT macro is executed, which links back to an external function utilizing GPT. A skeleton prompt is filled with a prompt to extract the necessary information using the user’s last response. The GPT function outputs a response with the GPT-3.5 API, which is preprocessed before being utilized in the dialogue to transition Emora to a different state. An example of the GPT prompt usage for Emora’s task categorization NLU will be provided in Appendix A.1.

In addition to macros linking to GPT functions, macros are also used for information transfer and database access for Emora. Once all of the information is extracted, macros for creating requests and messages are executed, using the extracted information to transfer the information across the database and to the designated recipients. If the tasks require the recipient to be notified, a notification will be pushed to the recipient’s device.

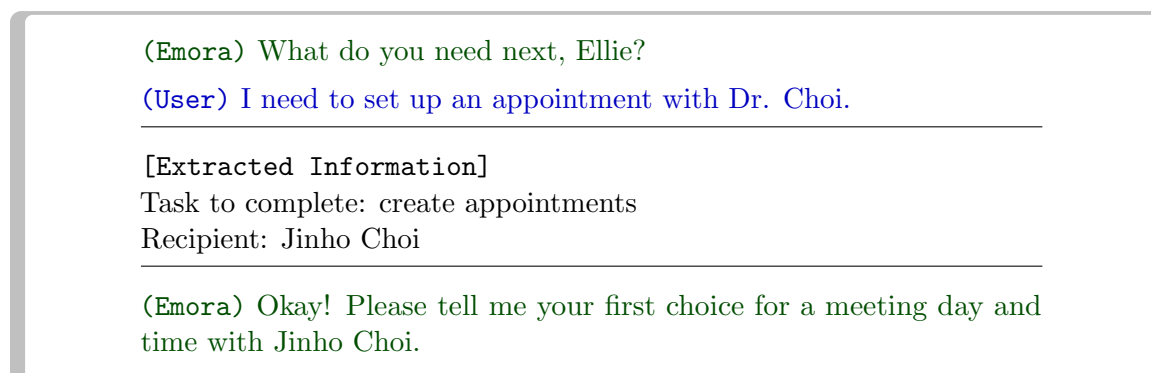


Figure 3.2: An example of a conversation between a user and Emora, where the user’s input is analyzed and the necessary information is extracted. Between the user and system transition, what is extracted from the user’s input using GPT is outlined.

Figure 3.2 demonstrates an example of such input extraction. In it, Emora is in

the 'hub' state, in which she is able to categorize tasks based on user input. The user requests to make an appointment with another person. The GPT function is executed for input extraction, first extracting the task to complete as making appointments. Emora then moves to the state in which appointments are made, and the GPT function is again executed, this time extracting the recipient of the appointment request. Once the name is extracted, Emora transitions to the next conversation state, where she requests for the meeting day and time with the recipient.

3.2 Database

For seamless information exchange and facilitation of communication and task execution, MongoDB is employed to store user information. The database stores user IDs and associated information, such as verification of whether they accepted the Terms of Service. It is important to note that the user ID that is stored is users' Emory IDs, and thus is unique. This helps the chatbot identify different users, even if users have identical first and last names.

In addition to user information, Emora has inboxes for storing users' appointment requests and messages, as well as groups' feedback questions and responses. For each user, there are three inboxes. The first stores appointment requests and notifications for confirmation, rescheduling, and cancellation of appointments (see Section 3.3.2). The second stores messages, including announcements (see Sections 3.3.3 and 3.3.4). The third inbox stores feedback, in which if the user is the leader of a group, the inbox consists of questions that they asked to a group and associated answers to the question.

3.3 Features

As of the time of writing, the Emora Assistant Bot handles seventeen unique tasks, each associated with administrative functions.

3.3.1 Names and Hub

When first conversing with the chatbot, Emora will ask the user what their name is and what they wish to be called. The first response is recorded into the database as their official name, but the second response may be customizable, allowing users to let Emora call them whatever name they wish.

This name calling function is active at all times, meaning users may switch what they wish to be called. If requested that they wish to be called a different name, the Assistant Bot's subsequent responses will include the requested name. In the case of providing multiple names, Emora will randomly use one of the names given to respond to the user.

After confirming what the user wishes to be called, Emora enters the "hub" conversation state. This is the state of the chatbot in which Emora is able to categorize various tasks that the user requests and transition to the appropriate state to resolve it. It is important to note that while a multitude of task categorization is possible, it is only possible once entering this "hub" state. If Emora is in the middle of a conversation state where she is trying to resolve a task, no other task can be inputted until that specific conversation state is exited.

3.3.2 Appointments

Emora facilitates appointment scheduling between users, managing details including the date and time, location, and reasons for meeting. Completing the extraction of such details will allow Emora to send the request to the recipient. By default,


```

(User) hi
(Emora) Before we begin, please provide your full name.
(User) Ryne Waters
-----
[Extracted Information]
Full name: Ryne Waters
-----
(Emora) Thank you Ryne Waters. Welcome! How should I call you?
(User) just ryne is fine
-----
[Extracted Information]
Names to call: [Ryne]
-----
(Emora) Nice to meet you, Ryne. What do you need next, Ryne?

```

Figure 3.3: An example of an initial conversation between Emora and a mock user with the name calling function. Between each user and system transition, what is extracted from the user’s input using GPT is outlined.

appointment requests are created with a length of 30 minutes, but this can be adjusted based on user request. Users can also provide alternative time requests to accompany the original message request.

Upon both parties’ agreement to meet, Emora will insert the decided meeting date and time into each of the users’ personalized schedules. Subsequent appointment suggestions — made by any user to the two users with the scheduled appointment — that conflict with this appointment are addressed.

If the recipient disagrees with the requested appointment time, Emora will request dates and times for when the event can be rescheduled. This will occur back and forth between the two users until a date and time in which both parties can meet is decided.

Any pending appointment requests can be edited or cancelled by the sender, and any pending appointment in which its date has passed can be rescheduled by both the sender and the recipient.

Figure 3.4 demonstrates an example conversation, in which a request for making an appointment is handled.

(User) Great, I'd like to make an appointment with Prof Smith

[Extracted Information]

Task to complete: create appointments

Recipient: John Smith

(Emora) Okay! Please tell me your first choice for a meeting day and time with John Smith.

(User) Tomorrow at 5:20

[Extracted Information]

Appointment date: 10/05/2023 05:20 PM to 10/05/2023 05:50 PM

(Emora) Ok, I think I got it. Is this what you meant?

10/05/2023 (05:20 PM) to 10/05/2023 (05:50 PM)

(User) yup

[Extracted Information]

Affirmation (Yes/No): Yes

(Emora) Awesome, I'll see if they are free. And where do you want to meet with John Smith?

(User) In his office

[Extracted Information]

Appointment Location: his office

(Emora) Ok, I will mark the location as his office. Would you like to include another alternative time to meet?

(User) no

[Extracted Information]

Affirmation (Yes/No): No

(Emora) Understood, that's it. Please tell me your reason for wanting to meet with John Smith.

(User) Discussing thesis things

[Extracted Information]

Appointment Reason: Discussing thesis

(Emora) Thank you! I'll send off your request:

Meeting, Haruto Suzuki and John Smith: 10/05/2023 (05:20 PM) to 10/05/2023 (05:50 PM) at his office. (Discussing thesis)

Figure 3.4: An example of Emora handling an appointment creation request. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

3.3.3 Messages

Users can request Emora to send messages to other users. This is a one-to-one messaging system, and cannot be applied to multiple people unless in a group (see Section 3.3.4). These messages, while maintaining the user's original contents, are also summarized using GPT-3.5. Thus, recipients will receive the original message that the sender sends, along with a summary of the message.

If requested, messages can be made anonymous, in which the sender's name is obscured when the message is sent to the recipient. However, it is important to note that the anonymity is maintained externally; messages still have the identity of the original sender intact within the database framework. This is to prevent any abuse of the anonymity system and ensure user safety through the chatbot.

While previously, appointments (see Section 3.3.2) and messages were stored in two separate inboxes, this had led to an overlap in components for checking different inboxes, and thus Emora had tendencies to miscategorize the two inbox-checking tasks. As a result, as of March 13, 2024, the two inboxes have been merged into one. When checking for updates, Emora will first check for miscellaneous messages before checking for appointment requests (see Figure 3.6).

Figure 3.5 demonstrates a conversation in which a message is sent to a desired recipient.

3.3.4 Groups

Emora supports group creation, where users may be added via name or ID. The creator of the group is regarded as the leader, and the users added to the group are regarded as members. Each group are uniquely identified by their name, and as consequence to this, multiple groups of the same name cannot be created.

Regardless of leader or member status, users can check what groups they are in, where it is provided to them in a list of group names with the corresponding group

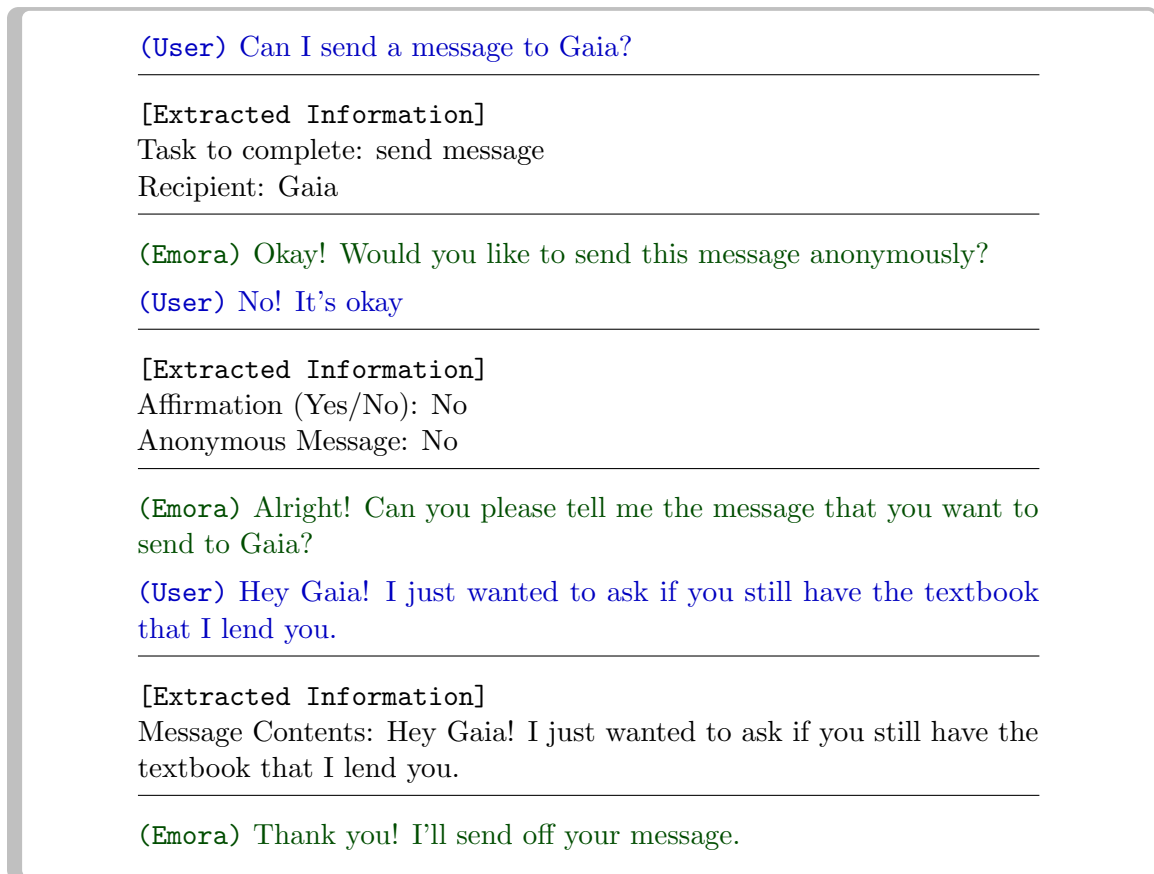


Figure 3.5: An example of Emora handling a message sending request. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

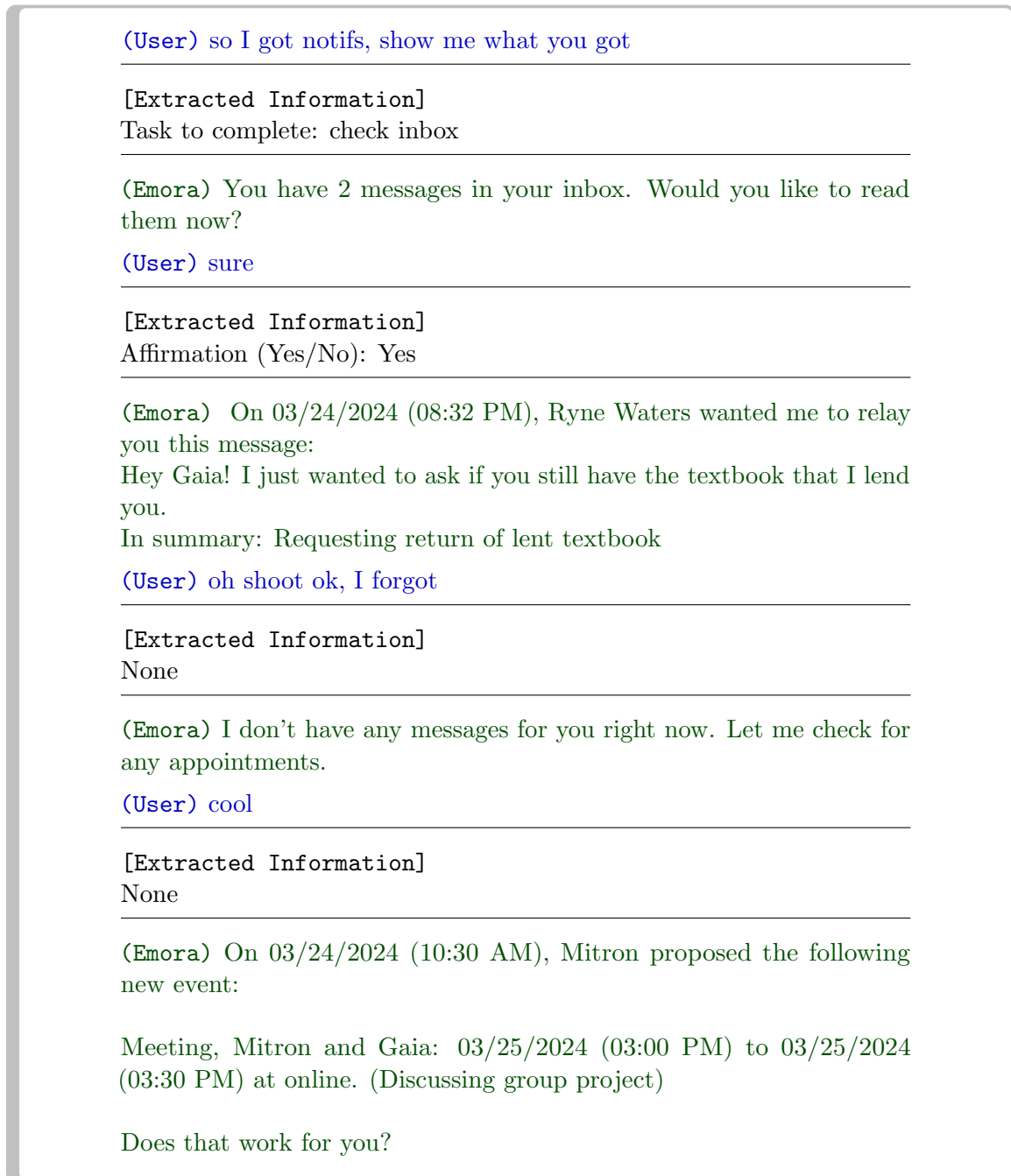


Figure 3.6: An example of Emora handling an inbox checking. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

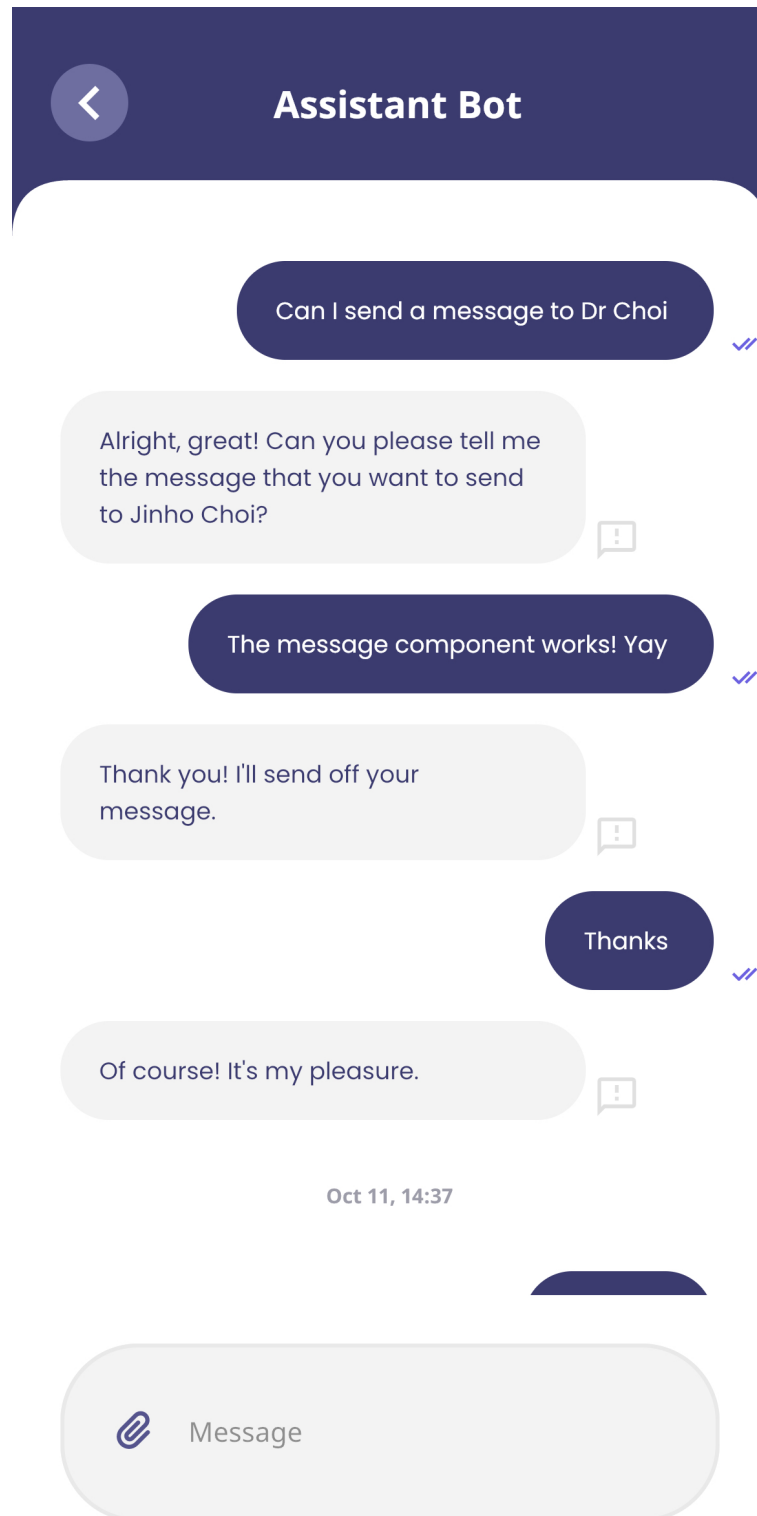


Figure 3.7: A screenshot of the Emora application, detailing a conversation for sending a message.

leader. Users can also check what members are in a group by specifically requesting for the identities of people who are in a specific group.

Figures 3.7, 3.8, and 3.9 all illustrate conversations in which groups can be made, and the groups that a user is in can be checked. It is important to note that checking the members of a group is a separate task, and thus Emora must be in the 'hub' state to receive the request and execute it.

Leader Functions

Leaders of groups have control over the creation of the group, in which if they choose to, they may delete the group that they have created. When the deletion is confirmed, the group and its associated information is removed from the database.

As extension to Section 3.3.3, leaders are able to send messages to members of their groups, identified as announcements. When confirmed, the message that the leader wishes to send to their group is stored in each of the members' inboxes until the inbox is checked by the members.

Finally, leaders are able to request answers to specific questions from their group, identified as requesting feedback from members. Once sent to the members, leaders are able to check for whether members have responded to the feedback question. GPT-3.5 abridges all of the responses into a cohesive summary if responses are requested.

Member Functions

As members of groups, users receive feedback questions from the group's respective leader and are able to respond to the question. While the responses are stored to be abridged into a summary, they may choose to respond anonymously. In addition, they may choose to leave the group that they have been added to. They cannot rejoin the group that they have left, and thus may have to request the leader for who can choose to delete the group. The users within the group are regarded as members, and while



Figure 3.8: An example of Emora handling a group creation request. Between each user and system transition, what is extracted from the user's input using GPT is outlined.



Figure 3.9: An example of Emora handling a group identification request. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

they cannot edit the group, they may choose to leave the group.

Figures 3.10, 3.11, and 3.12 demonstrate conversations between Emora and two different users. In it, the leader requests Emora to send a feedback request to the group, in which the member receives the request and gives feedback. When the leader checks for updates on the feedback question, the output is a summary by GPT that abridges the responses received so far.

(User) I'd like to request feedback from my LING 200 class

[Extracted Information]
 Task to complete: request feedback
 Group: LING200

(Emora) Alright! Can you please tell me the feedback question that you want to send to LING200?

(User) I'd like to request feedback for the midterm that everyone took. Was it too easy, difficult, or just right? If there're any questions that you felt were confusing or difficult, let me know.

[Extracted Information]
 Feedback Question: I'd like to request feedback for the midterm that everyone took. Was it too easy, difficult, or just right? If there're any questions that you felt were confusing or difficult, let me know.

(Emora) Thank you! I'll send off your message.

Figure 3.10: An example of Emora sending feedback requests to groups. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

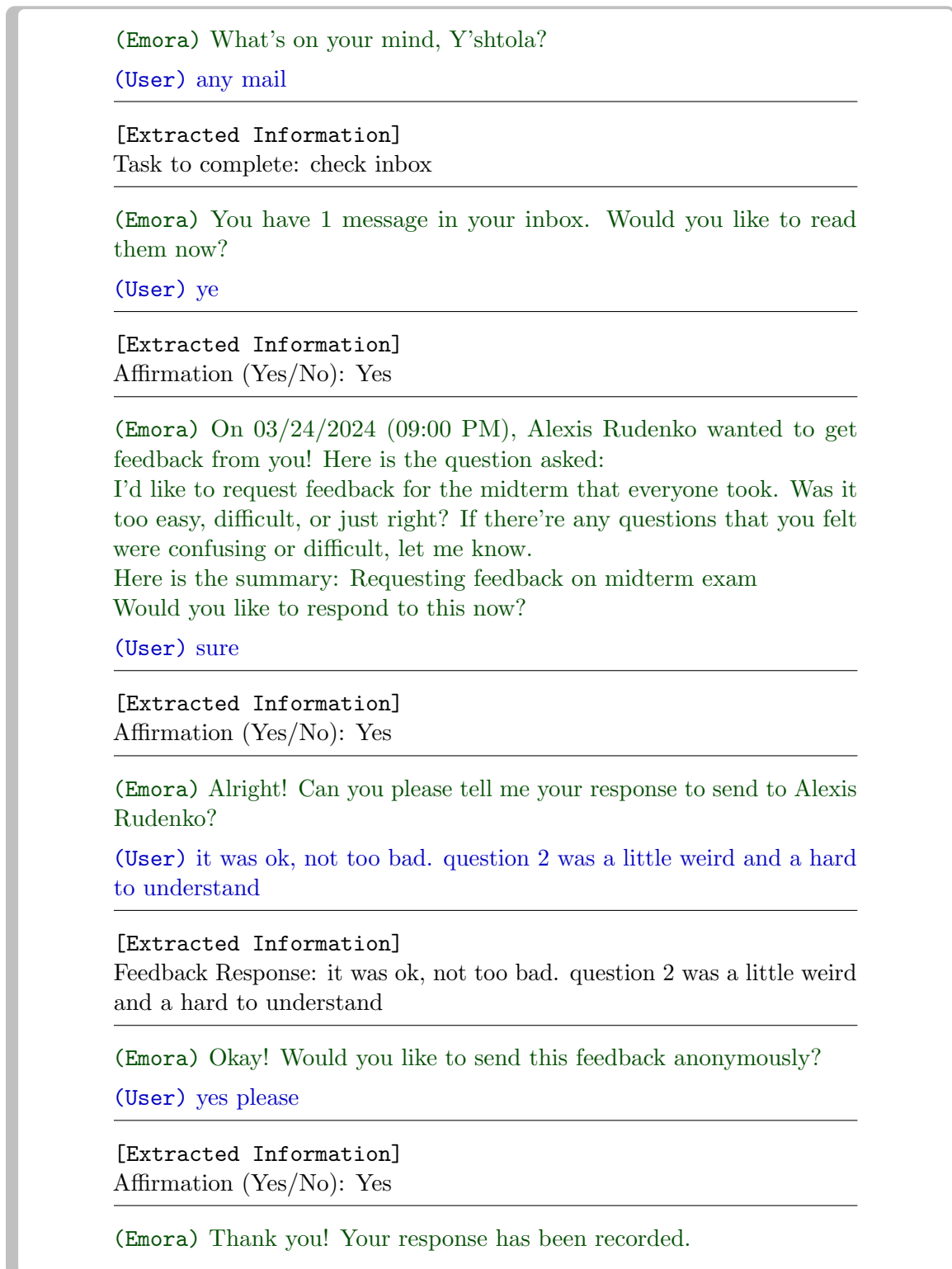


Figure 3.11: An example of Emora recording feedback from group members. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

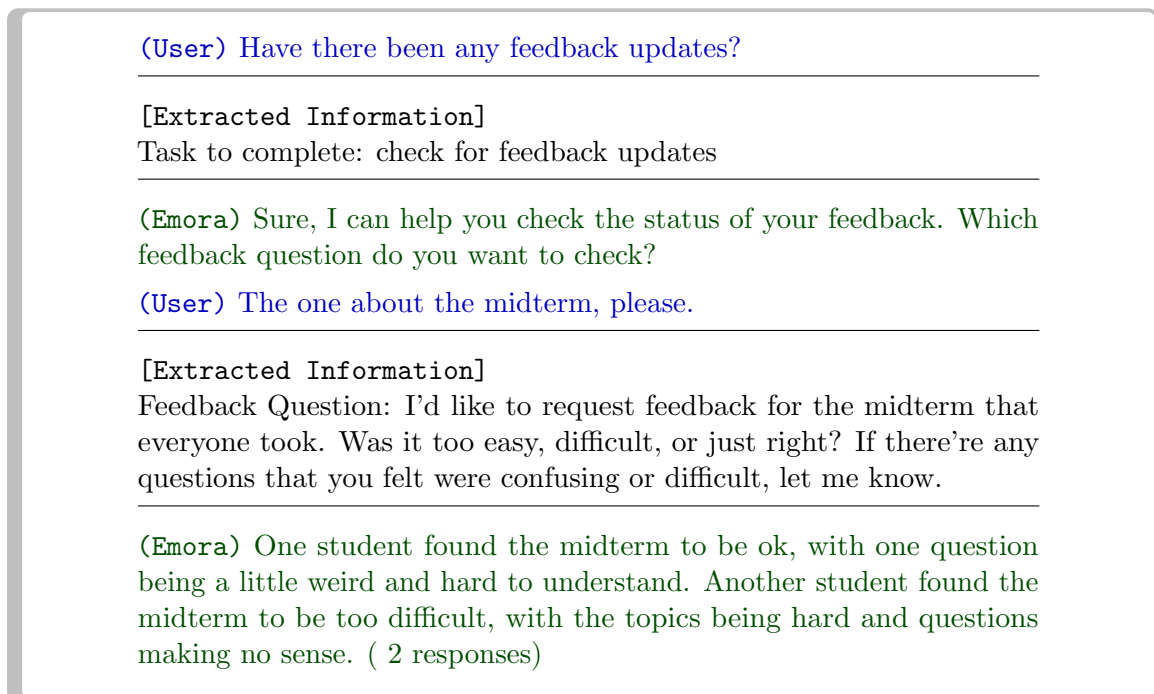


Figure 3.12: An example of Emora handling feedback checks and providing a cohesive summary. Between each user and system transition, what is extracted from the user's input using GPT is outlined.

Chapter 4

Experiments

Since the Emora Assistant Bot is a chatbot intended for humans, it is imperative that the performance of intended functionalities and appropriate responses are evaluated. To that end, the following evaluation method is implemented.

4.1 GPT Evaluation

While human-based evaluation may most accurately demonstrate a real-life environment, recruiting a large sample to thoroughly evaluate the functionalities of Emora is inefficient from a time and resource perspective. Consequently, a different approach to simulate a classroom setting is initiated. In this classroom simulation, GPT-3.5 is utilized to act as all of the users talking to Emora. The conversations between the GPT simulation and Emora is generated and recorded. Following the evaluation generation, human annotators observe the conversation between the GPT simulation and the assistant bot, and evaluate both sides' performances.

4.1.1 Evaluation Framework

To facilitate the possibility of multiple people talking to one another, the AsyncDialogue framework is used as the foundation of the simulation. This framework, developed by a graduate student while working in collaboration on the Emora Course Assistant project (see Section 2.3.1, was originally developed for internal testing of multiple users and proper transfer of information by Emora.

AsyncDialogue allows for conversations between multiple users to occur in one execution of the bot program. The framework first accepts the bot and mock database parameters, which are required for the information to pass between multiple mock users. Then, conversations between one user ID and the bot may occur, with user and system turns occurring one after the other. When the program requests a user switch, or the event in which another "user" wishes to talk to the bot, the framework saves the current conversation between the previous user and bot to the previous user's ID. Then, it loads the new user ID into the framework, as well as the conversation that the new user previously had with the chatbot. This way, conversation between the second user and the chatbot will continue, and through the framework, the conversations can again be changed to new users or users in which conversations with the chatbot have already been initiated.

4.1.2 User Profiles

All user profiles that is simulated by GPT-3.5 is generated through ChatGPT, the online version OpenAI's LLM. Each user profile contains the following information:

1. First and last name.
2. ID for use in the AsyncDialogue. To accurately simulate the environment of the application, each ID is unique from one another and typically contains the user's last name.

3. Occupation, or whether they are a student or professor.
4. Personality
5. Studying tendencies
6. Online tendencies. This was generated to help diversify the responses that each user would provide to Emora.
7. Class information. This includes the class name, professor name, and students or classmates' names depending on if the user is a student or professor.
8. Class history. This is a dictionary that contains the material that was covered in the class, as well as the user's history of their learning or instructing progress. For time restriction purposes, experiments have been conducted with the class having received one week of instruction.
9. Current material. This is the current material in the class that is being covered, and the point in time in which the simulated class is in. For time restriction purposes, experiments have been conducted with the class being in the second week of instruction.
10. To do. This is the task that the user wishes to complete through Emora. Since Emora can complete a limited number of given tasks, the task scope has been narrowed to be generated within the tasks that Emora can complete.
11. Chat history. This is the chat history that the user has with Emora. It is saved whenever the simulation switches to a different user and loaded into the response generation prompt whenever the simulation switches to the specific user. After each subsequent user and system response, the history is updated with the conversation.

12. Last response. This stores Emora's very last response to the user before the conversation is saved to the user profile. This is saved separately from the history to be highlighted in the response generation prompt. After each subsequent user and system response, the last response is updated with Emora's new last response.

Each user profile is saved as a JSON file, with its name being the user ID. This is to facilitate loading the user's profile into the simulation when switching users to a different ID.

4.1.3 Simulation Process

Since the mock database is reset with each execution of the program, each iteration of the GPT simulation is initialized prior to continuing the conversation. In each initialization, a set user turn is inputted into the dialogue framework, in which the Assistant Bot responds with an inquiry of the user's name. Then, the simulation responds with the selected user's name. After a subsequent bot turn, the framework switches to a different user id, and the process is repeated until all of the given users are initialized. After the initialization, the simulation switches to a given starting ID, and the simulation begins.

To prevent any overloads of the GPT-3.5 API, each evaluation iteration was limited to a set number of turns ($n=500$), inclusive of both system and user turns. Thus, a system and a user turn would add to two turns total. The simulation would be executed and would be terminated if the number of turns has been reached, or all of the given users' tasks and notifications have been resolved through the bot. In the latter case, a system message, `All tasks and notifications resolved.`, will be recorded.

When a given user is selected to converse with Emora, the user's history is first loaded into a prompt formatted for GPT-3.5's response. Then, the prompt is inputted

to GPT, and a response in adherence to the user's character will be outputted. The simulation then submits the response as a user turn and generates a subsequent system turn. The history and Emora's last response for this given user is updated with the two turns that have been generated.

Afterwards, a second function utilizing GPT is executed. This function examines the conversation between the selected user and Emora and determines whether any notifications or tasks that this user had prior to the conversation update has been resolved. The response is outputted in a dictionary format of

$$\{\text{"user ID"} : \{\text{"notifications":}x, \text{"tasks":}y\}\}$$

where x and y are the value of notifications and tasks determined by GPT. This result is used to update a file used to store all users' notifications and tasks.

Finally, a third function utilizing GPT is executed. This function, like the second function, examines the conversation between the selected user and Emora. Unlike the second function, however, this determines whether the conversation between the user and chatbot has reached its natural end or is repeating. The output is either a YES or NO.

If the result is NO, the process repeats with the current user until the result is YES.

If the result is YES, a user different from the currently conversing user is selected from a list of provided users in the simulation. This selection is determined through a random sampling of the user list, where each user is weighed by the number of tasks that they need to complete and the number of notifications that they currently have. Since in a real life setting, people tend to check notifications that are pushed to their devices, the number of notifications are given a higher weight compared to the tasks.

Each user's weight is calculated through the equation below:

$$\frac{(n_s * 2.5) + (t_s * 1)}{\sum_{i=1}^N (n_i * 2.5) + (t_i * 1)}$$

where n_i and t_i respectively correspond to the number of notifications and tasks of a user i , and n_s and t_s respectively correspond to the number of notifications and tasks of a specific user s . N corresponds to all users in the list of users, with exception of the user that last was having a conversation with Emora.

Once each users' weights have been determined, the random sampling outputs a new user to switch to. The Async Dialogue framework switches to the new user ID's conversation, and the new user's information is loaded before the process repeats.

The GPT prompts used to generate simulation responses, as well as determine the number of tasks and notifications and evaluate whether a conversation has ended or is repeated will be provided in Appendix A.2.

4.2 Evaluation Criterion and Metrics

Each evaluation is manually reviewed by human annotators, given the following criterion to evaluate both the GPT simulation and the Assistant Bot. Since the initialization portion includes a manual adjustment, the initialization section of the simulation is not evaluated and its success rate is not included in the final metrics.

4.2.1 Emora Assistant Chatbot

Since the Emora Assistant Chatbot is a task-oriented chatbot, the criterion for evaluating this chatbot is as follows.

First, whether Emora successfully completed the user's task is evaluated. Each success is identified when Emora successfully completes the task among the seventeen that she can do, or if the user requests a task that Emora cannot handle and she responds appropriately. If the user requests for multiple functions within a turn,

each requested function is regarded as an individual and whether Emora responds appropriately is evaluated. If Emora responds with a question that inquires the user whether they wish for assistance the task that they want handled, this turn will not be considered as a failure because Emora still successfully categorizes the task.

The complete success rate, or success rate with the STDM fails (**SWS**), is measured as a proportion of the number of tasks that Emora successfully completed to the total number of tasks that was requested to Emora in the given evaluation (**Tasks**).

It is important to note that some of the task failures are rooted in the state machine-based framework’s limitations, which can only be resolved through the use of a new chatbot framework. As a result, the number of tasks failed due to the STDM (**STDM**) is counted, and the success rate excluding the STDM fails (**SES**) is also calculated. This success rate is measured as a proportion of the number of tasks that Emora successfully completed to the total number of tasks that was requested to Emora, excluding the tasks that she failed to accomplish due to framework limitations.

Second, each turn generated by Emora is scrutinized to evaluate whether Emora’s response successfully demonstrates natural language understanding. This is divided into three categories.

The first category determines whether the Emora Assistant Chatbot responds appropriately in context to the user’s last statement. If any of Emora’s turn demonstrates an abrupt transition from the user’s last response, then it is determined that Emora responded inappropriately. The inappropriate response rate (**IR**) is calculated as the proportion of the number of inappropriate responses to the total number of system turns in the given evaluation.

The second category determines whether the Assistant Bot extracted the correct information from the user. This includes any form of task categorization that Emora conducts, as well as information extraction within the individual tasks. The improper input extraction rate (**IIE**) is calculated as the proportion of the number of turns

Emora incorrectly extracted the user’s input to the total number of system turns in the given evaluation.

The third category, similar to the criterion from the task criterion above, determines whether Emora’s response fails to demonstrate proper natural language understanding or information extraction due to an STDM limitation. Due to the nature of the Emora STDM, one of its limitations is that if the conversation state transitions into a specific conversation, the chatbot cannot exit from the state until the conversation is fully resolved. In addition, there are errors that occur due to its limitation of being unable to process special characters such as newlines. Thus, any inappropriate responses due to such STDM limitations are recorded in a category separate from the first two. The fail rate due to STDM limitations (**STDM**) is calculated as the proportion of the number of turns Emora failed to respond appropriately due to an STDM failure to the total number of system turns in the given evaluation

Finally, the cumulative success rate in which Emora demonstrates utterance-based natural language understanding, or success rate with the STDM limitation fails (**SWS**), is calculated as the proportion of the total number of turns Emora successfully demonstrated natural language understanding to the total number of system turns in the given evaluation. The success rate excluding STDM limitations (**SES**), will also be calculated, in which it is the proportion between the total number of turns Emora successfully demonstrated natural language understanding to the total number of system turns in the given evaluation, excluding the turns in which she failed to understand due to an STDM limitation.

4.2.2 GPT Simulation

The GPT simulation is evaluated based on the criterion of whether there was an instance of failure in each user turn. This is including, but not limited to, hallucinations by the simulation, inappropriate responses given Emora’s last response, and irrelevant

repeating statements given to the user's previous responses. If the repeating statement is in relevance to the task that the user wishes to have completed, it is not considered irrelevant.

The success rate of the simulation (**GPT**) is determined by the number of turns in which the simulation did not fail, and will be compared to the total number of turns in which the simulation speaks in a given evaluation. This includes all of the users that have been simulated in the evaluation.

4.3 Results and Discussion

This section displays results of the evaluations generated from the GPT simulation. Each iteration of the evaluations demonstrates conversations between multiple mock users and Emora, with the cumulative turns being 500. This means that in each evaluation, 250 user and 250 system turns occurred.

Two different variables were changed. First, the number of students in a given classroom were adjusted to observe if the GPT simulation would stay consistent with larger populations. For this experiment, a classroom of 10 (1 v. 10) and 20 students (1 v. 20) were simulated, with 1 professor staying consistent. As a result, 20 student profiles and 1 professor profile are created, with 10 of those student profiles being used for both the 1 professor to 10 students and 1 professor to 20 students experiments.

Afterwards, the number of professors and classes were changed. Due to the restriction of time, only two total classes were simulated, with 2 professor and 20 total student profiles were used (2 v. 20). As this was derived from the 1 professor versus 20 students simulation, one professor maintained a class of 20 students, while the second professor maintained a class of 10 students, picked randomly from the population of 20 students.

Following the first set of evaluations on 1 v. 10 and 1 v. 20 experiment conditions,

adjustable flaws in both the simulation and the Emora Assistant Chatbot were revealed. As a result of this discovery, both the GPT simulation and Emora Assistant Chatbot received adjustments.

The GPT simulation received some adjustments through additional examples in few-shot prompt engineering, specifically in the function for resolving tasks and notifications of a given user. In addition, while the first version’s weight for notifications was a 1.5 multiplication factor, this second version gave the notifications a greater weight by a 2.5 multiplier. Meanwhile, Emora’s NATEX and responses in components and global transitions were adjusted in accordance with the errors found through the first set of evaluations. The GPT functions used within Emora for task categorization was adjusted as well, with more specific examples added to the feedback-related tasks.

As the subsequent results are from different simulations, the first iteration of the simulation will be referred to as Version 1, while the recently updated version of the simulation will be referred to as Version 2. Evaluations for multiple classes was only generated using Version 2 of the simulation.

4.3.1 Results from Version 1

Tasks				Emora Utterance					GPT
Tasks	STDM	SWS	SES	IR	IIE	STDM	SWS	SES	GPT
162	135	14.9	88.9	0.4	2	54.4	43.2	94.7	82.4
164	60	61	96.2	0.4	1.6	25.6	72.4	97.3	69.6
116	63	41.4	90.6	0	25.2	26.4	48.4	65.8	81.6
145	90	36.6	96.4	0	2	48	50	96.2	84
88	60	27.3	85.7	0	27.6	24	48.4	72.4	79.6
53	20	56.6	90.9	0.4	30.4	7.6	61.6	63.7	66.4
201	62	63.2	91.4	0.4	5.6	24.8	69.2	92	94
100	34	44	66.7	0.8	31.2	13.6	54.8	63	78.4
90	27	66.7	95.2	1.2	14	10.8	74	83	86
124	61	45.7	89.1	0.4	15.5	26.1	58	80.9	80.2

Table 4.1: Rates of Success by Emora and GPT Based on 1 Professor, 10 Student Profiles (Simulation Version 1). The bottom row displays the average of the column’s values.

Tasks				Emora Utterance					GPT
Tasks	STDM	SWS	SES	IR	IIE	STDM	SWS	SES	GPT
91	40	48.4	86.3	0.4	3.6	16	80	95.2	73.2
110	47	40	69.8	1.2	11.2	18.8	68.8	84.7	86
112	52	49.1	91.7	0.8	4.4	20.8	74	93.4	85.2
101	43	52.5	91.4	0.8	9.2	17.2	72.8	87.9	84.8
139	68	48.2	94.4	1.6	5.2	27.6	65.6	90.6	88
111	50	47.6	86.7	1.0	6.7	20.1	72.2	90.4	83.4

Table 4.2: Rates of Success by Emora and GPT Based on 1 Professor, 20 Student Profiles (Simulation Version 1). The bottom row displays the average of the column’s values.

Tables 4.1 and 4.2 demonstrates the success rate of Emora completing tasks and understanding the user. Immediately, it is clear from both results that the limitations from the STDM heavily impact the success rate in which the Assistant Bot resolves tasks and understands the users. With the exclusion of STDM limitation fails, the task success rate increased by approximately 20-70%, while utterance success rates increased by approximately 2-50%.

Overall, rates from the 1 v. 20 evaluations seemed to fluctuate less than rates from the 1 v. 10 evaluations, especially in respect to the Emora Utterance **IIE** and **SES** rates. While it is important to note that there are less iterations of evaluations generated for the 20 students-variable compared to the 10 students-variable, the consistent, comparable rates may suggest that the GPT simulation is able to generate comparable, and more consistent, evaluations with a higher number of mock users.

One important aspect to highlight is the low **IR** rate that Emora displays, of which the rates range from 0-1.6% across both sets of evaluations generated. This low rate may be attributed to the factor that the Assistant Bot is a strictly task-oriented bot, thus the conversation flow would be more in relation to the extraction of information rather than a wide variety of topics like a social chatbot. Thus, much of Emora’s responses would be oriented towards information extraction and task management, allowing for less inappropriate responses to be generated.

On the other hand, the rate of **IIE** fluctuated, ranging from 2-31.2% for the 10 student-variable and 3-11% for the 20 student-variable. Similarly, the success rate of the GPT simulation also fluctuated, ranging from 66-94% for 10 students and 73-88%. The variable of a larger number of students with a fixed number of cumulative turns may have resulted in a smaller yet slightly lower percentage range, yet there is still a relatively wide variance in GPT simulation performance.

While it is difficult to pinpoint the precise reasons as to why these rates differ in wide ranges, it may be attributed to the flow of conversation that the simulation and the Assistant Bot has. As the simulation progresses and various users' needs are resolved, the GPT simulation outputs responses that are of the same contents as the user's previous response, and unrelated to requesting a new task. As Emora responds accordingly, the simulation falls into a loop of the simulated user repeating itself and Emora responding to the user, resulting in an evaluation of a higher simulation error.

Similarly, there are points in the conversation in which the simulated user provides information for a task in which the Emora Assistant Chatbot incorrectly receives, whether by task miscategorization or a mistake in the input extraction. Since the chatbot did not extract the correct information in accordance with the user's last response, the GPT simulation repeats itself, providing the information once more. This results in another loop of the simulated user repeating itself and Emora incorrectly receiving the provided input. In this situation, the simulation is not evaluated to have failed due to the user repeating themselves to complete a task (see Section 4.2.2), but the inappropriate response rate on Emora's turn-based evaluation increases due to the repeated errors in information extraction.

While in total, 10 iterations of the 1 v. 10 simulation was generated, Table 4.1 displays nine. This is because there was an iteration of the evaluation in which the turns for Emora and the GPT simulation were both 94, shorter than the 250 turns that the other iterations produced. This was due to the evaluation terminating before

the threshold of 500 cumulative turns, as it had determined that all of the tasks and notifications for all users had been resolved. However, an observation of the generated conversation revealed that not all of the users’ tasks and notifications were properly resolved by the chatbot, insinuating a fail in the simulation’s task and notification resolving function. As a result, this iteration of the evaluation generation has been removed from the final results.

Due to a relatively large fluctuation in the success rates for both tests, the simulation’s prompts and functions were reviewed using the generated evaluations. Using the common errors that the simulation and Emora made, both were adjusted, and additional tests were run using Version 2 of the GPT simulation.

4.3.2 Results from Version 2

Tasks				Emora Utterance					GPT
Tasks	STDM	SWS	SES	IR	IIE	STDM	SWS	SES	GPT
77	26	58.4	88.2	1.2	8.4	10.4	80	89.3	83.2
101	57	42.6	97.7	2.8	2	22.8	72.6	93.8	64.4
127	28	76.4	98	0.4	3.2	15.6	80.8	95.7	87.6
112	45	42	70.1	1.6	12	23.2	63.2	82.3	80.8
78	29	59	93.9	0.8	3.2	12	84	95.5	60.4
99	37	55.7	89.6	1.4	5.8	16.8	76.1	91.3	75.3

Table 4.3: Rates of Success by Emora and GPT Based on 1 Professor, 10 Student Profiles (Simulation Version 2) The bottom row displays the average of the column’s values.

Tables 4.3 and 4.4 display the results of the GPT Simulation Version 2, evaluated on 1 professor and 10 and 20 student profiles respectively. Due to time restrictions, only five evaluations of 500 cumulative turns were generated for each variable set.

It is important to note that in comparison to Version 1’s results from Tables 4.1 and 4.2, the rates are marginally more consistent, particularly when evaluating the utterance-level successes. The **IIE** rates for 10 students have an approximate 10% range of fluctuation between one another, while the first version had an approximate

Tasks				Emora Utterance					GPT
Tasks	STDM	SWS	SES	IR	IIE	STDM	SWS	SES	GPT
103	37	60.2	91.2	2.4	4.4	18	75.2	91.7	83.6
125	40	67.2	98.8	0.4	3.2	35.2	61.2	94.4	85.2
105	41	56.2	92.2	0.8	4.8	16.8	77.6	93.3	90
116	36	64.7	93.8	1.6	3.6	14.4	80.4	93.9	90
116	46	50	82.9	1	6	20.8	72.4	91.4	81.6
113	40	59.7	91.8	1.2	4.4	21	73.4	92.9	86.1

Table 4.4: Rates of Success by Emora and GPT Based on 1 Professor, 20 Student Profiles (Simulation Version 2) The bottom row displays the average of the column’s values.

28% range of fluctuation. For 20 students, Version 2 demonstrated a 3% difference while Version 1 results had an 8% difference. The rates for Emora Utterance’s **SES** and **GPT** are also less variable for both sets compared to the first version, demonstrating that the changes made to the simulation framework helped improve the GPT simulation to perform more consistently.

Overall, task performance for Emora across both evaluation sets are high, with an 89.6% average **SES** rate for 10 students and 91.8% average **SES** rate for 20 students. Average utterance performance is higher for both sets, with the utterance **SES** rates being 91.3% and 92.9% respectively. GPT simulation success rate is lower than the success rates for Emora, though it still boasts a relatively high performance; the average success rate for 10 students is 75.3% and the average success rate for 20 students is 86.1%.

Similar to the results from Version 1, the 20-student evaluation set displays results that are less variable and — on average — higher than the 10-student evaluation set. While the 10-student evaluation set displayed improvements in consistency through the adjustment of the simulation, there is still a larger variance in the results compared to the 20-student evaluation set. This may further demonstrate that the GPT simulation not only performs with increasing accuracy with additional mock users, but may perform more consistently as more users are introduced to the simulation.

Multiple Classes Variable Results

Tasks			Emora Utterance					GPT	
Tasks	STD M	SWS	SES	IR	IIE	STD M	SWS	SES	GPT
118	40	63.6	96.2	1.2	2.8	21.2	74.8	94.9	80
108	49	50	91.5	1.2	3.2	21.2	74.4	94.4	84
97	37	59.8	96.7	1.2	3.2	19.2	76.4	94.6	79.6
95	33	60	91.9	0.8	7.2	16.4	75.6	90.4	80.4
121	58	49.6	95.2	0.4	2.8	23.2	73.6	95.8	81.2
106	43	56.6	94.3	1.0	3.8	20.2	75	94	81

Table 4.5: Rates of Success by Emora and GPT Based on 2 Professors, 20 Student Profiles (Simulation Version 2) The bottom row displays the average of the column’s values.

Table 4.5 illustrates the results of evaluations using Version 2 of the simulation, but with two professors and 20 students. The results for evaluating the Emora Assistant Bot is more consistent than the previous evaluation sets, with both the tasks’ and utterances’ **SES** rates having about a 5% range. The GPT success rate is also more consistent, with the range also being around 5%.

The average **SES** rates for the Assistant Bot is also higher than the previous evaluation sets. For tasks, the average **SES** rate was at about 94.3%, while for utterances, the average **SES** rate was approximately 94.0%. Both rates are an increase from the evaluations with the one class variable, suggesting that Emora is robust with handling requests from students and professors in multiple classes.

By contrast, the GPT simulation demonstrated a lower performance average compared to the 1 v. 20 evaluation set; the average success rate was at about 81.0%, which is lower than the 86.1% success rate that the 1 v. 20 GPT success rate boasted.

This may also be attributed to GPT variability and inconsistency, coupled with the variables of multiple users and multiple classes. The simulation must handle multiple user profiles, of which some students have multiple classes and tasks that may overlap. In addition to this, the simulation must determine if the users’ requested tasks are resolved, as well as transition points to switch to new users. As a result of

these multiple variables, an increased class count may have led to increased variability in GPT responses, which may have resulted in a lower success rate in the 2 v. 20 evaluation set compared to the 1 v. 20 evaluation set.

Despite it, however, the GPT simulation still demonstrates a relatively high success rate, and coupled with the high success rates of the Assistant Bot, is able to demonstrate a stable and consistent automatic chatbot evaluation.

Chapter 5

Analysis

This chapter takes a closer analysis at the results of the evaluations between the Emora Assistant Chatbot and the GPT simulation, specifically the errors that both Emora and the GPT simulation made. While some of the issues of both mediums have been categorized, this analysis will further divide the specific issues that the evaluations introduce, and bring to light the limitations of GPT-3.5 and the STDM framework that the chatbot utilizes.

For the analysis, evaluations from the 2 professor, 20 student-variable set were chosen, as the results are the most consistent, and the variables demonstrate a more complex, realistic environment than the variable of one class.

5.1 Analysis of Emora Errors

The Emora Assistant Bot's errors was divided into three categories: STDM limitations, improper input extraction, and inappropriate responses. Further error analyses was done on each of these error categories to determine the precise areas of improvement for the chatbot.

5.1.1 Inappropriate Responses

Out of all Emora Assistant Bot’s errors, the chatbot had the least amount of inappropriate responses. Since Emora is a task-oriented chatbot, many of her responses are strictly adhering to the task that is requested, thus the responses are often appropriate for the conversation.

However, there were occasions when Emora would give an inappropriate response to the conversation context. Out of the 12 total inappropriate responses given over the five evaluation sets, half of the responses were responses that ignored the user’s last response, and half of the responses were responses that did not fit into the context of the conversation. Figures 5.1 and 5.2 demonstrate examples of occurrences in which Emora responded inappropriate in terms of each category.

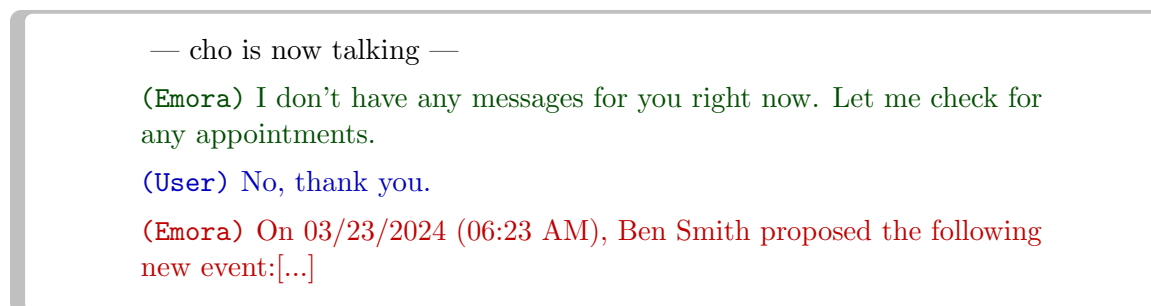


Figure 5.1: An example of Emora ignoring the User. The inappropriate response is in red.

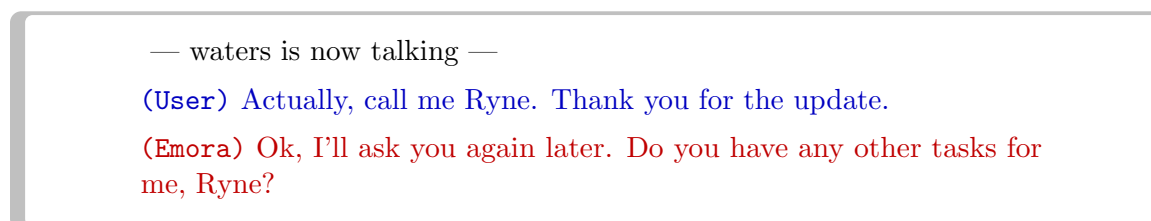


Figure 5.2: An example of Emora responding beyond the conversation context. The inappropriate response is in red.

Both types of inappropriate responses are tied to the STDN framework, in which the task that is requested to be resolved locks Emora into a linear conversation state and cannot shift to a different conversation, or handle different inputs. Thus, even when the user responds with a different yet related response, Emora’s subsequent

response may appear inappropriate to the conversation flow. While these response errors are also tied to STDN limitations (see Section 5.1.3), they are addressed separately due to the separate error category that they have been assigned through the experiments.

5.1.2 Improper Input Extraction

As a robust LLM, GPT is known for effectively extracting information, particularly through few-shot prompting. As a result, it was used to facilitate most of the NLU for the Assistant Bot, particularly all task categorization and information extraction tasks. While the success rate of input extraction was high, some responses that Emora generated was a result of improper input extraction. Out of the 48 responses that demonstrated improper input extraction, 46 of the turns were a display of incorrect task categorization, and two were a display of incorrect information extraction.

Since GPT has variability, it is possible that it led to unknown tasks being incorrectly categorized into tasks that Emora can do. Figure 5.3 demonstrates one such conversation, where a user requesting resources for research became categorized into that of sending a message to a different user. Both tasks are inherently different, yet it is possible that due to GPT variability, the task became incorrectly categorized.

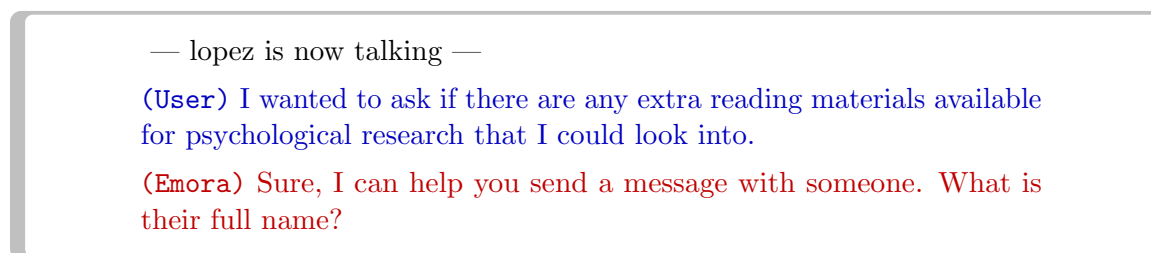


Figure 5.3: An example of Emora incorrectly identifying a messaging task within a request. The inappropriate response is in red.

Incorrect task categorization also occasionally occurred when a user demonstrated gratitude towards Emora, expressing a task that they would do on their own time, such as reach out to Emora (see Figure 5.4, or a task that the Assistant Bot had

already completed. In these cases, the GPT NLU cannot understand the nuance and incorrectly categorizes the task that was already done or unrelated to the chatbot and considers it a task to complete.

— patel2 is now talking —

(User) Thank you! I'll definitely reach out if I need any more help.
Have a great day!

(Emora) Sure, I can help you send a message with someone. What is their full name?

Figure 5.4: An example of Emora incorrectly identifying a messaging task within a statement of gratitude. The inappropriate response is in red.

Incorrect information extraction, a subcategory of improper input extraction, is when extracting particular information for a specific task, Emora extracts the incorrect information from the user's input. For example, names and locations may be incorrectly extracted and stored by Emora, such as in Figure 5.5. Rather than moving to a transition state where Emora expresses the requirement for a full name, the GPT NLU incorrectly extracted "Group" from the user's input and attempted to search for a name similar to it in the database. This led to the conversation state in which Emora said she cannot find a nonexistent name in the database.

— cho is now talking —

(User) Add all students to the CS101 Intro to CS Class group.

(Emora) I can't find Group in my records. What is the full name of the people with whom you want to put in your group?

Figure 5.5: An example of Emora incorrectly extracting information from a user's input. The inappropriate response is in red.

Incorrect information extraction may also be attributed to GPT variability, where on occasion, GPT may alter the extracted information rather than keeping the purely extracted text. The purpose for this may vary, such as changing the user's input, "Intro to CS," to "CS101" for cleaner format purposes. Thus, GPT's tendency for variability in extracting information may lead to the wrong or altered information

being extracted from user input, leading to incorrect information extraction.

5.1.3 STDM Limitations

Abbr.	Error Type	Description	%
SL	State Lock	Stuck in a linear conversation component	0.63
NI	NATEX Issues	Unintended NATEX NLU is captured	0.25
SC	Special Characters	Cannot handle certain special characters and raises an error	0.11
ST	Single Transition	Cannot handle multiple tasks at once, must choose one component	0.01

Table 5.1: Results of the STDM-based error analysis on the Emora Assistant Bot.

Errors due to STDM limitations made up a large part of Emora’s task and utterance-based errors; as mentioned in Chapter 4, the exclusion of STDM-based errors led to large increases in rates for both task and utterance successes. Table 5.1 illustrates the four error categories of errors based on 253 turns in which Emora demonstrated an STDM-based fails.

State Lock State Lock was one of the largest reasons to Emora’s turn and task-based fails. Since the state machine framework consists of a dialogue flow that often flows in a linear direction, it is difficult for a user response to transition the chatbot into a completely different component unless the transition is manually developed by the developer. In the case of a task-oriented chatbot such as Emora, many of the tasks that she can complete requires the user to provide the necessary input to execute said task. As a result, when a task is in the process of being executed, the conversation related to it must flow to its end state, where Emora completes the task requested, or canceled — which is handled by a global NATEX transition. Then, Emora will enter the ‘hub’ state, where she is able to categorize any input once more.

This issue often occurred during the start of the conversation and during conversation loops. The user always begins in the name calling module, where Emora asks the user what they wish to be called. However, many users in the simulation provided responses that either directly requested for a task, or asked for a task to be handled

after mentioning their name. Since at this stage of the conversation, Emora is still locked in the conversation state of the name calling component, she cannot take in the task that is requested. As a result, the task handling results in a fail.

— nakamura is now talking —

(Emora) Thank you Yuki Nakamura. Welcome! How should I call you?

(User) Just call me Yuki. Any updates?

(Emora) Nice to meet you, Yuki. What do you need next, Yuki?

Figure 5.6: An example of Emora not handling the check inboxes tasks due to the STDM State Lock (SL). The inappropriate response is in red.

State Lock was often the reason of certain user conversations resulting in endless loops. When a user entered a component’s dialogue flow due to an incorrect task categorization (see Section 5.1.2, the simulation would respond with a different task that they wish to have completed. However, because the conversation state was already locked into the component’s conversation states, Emora could not understand what the user wished to do and would be unable to respond appropriately as well as handle the requested task.

— lopez is now talking —

(User) I wanted to ask if there are any extra reading materials available for psychological research that I could look into.

(Emora) Sure, I can help you send a message with someone. What is their full name?

(User) Actually, I wanted to ask if there are any extra reading materials available for psychological research that I could look into.

(Emora) I couldn’t parse a name from what you said. What is the full name of the person with whom you want to send a message?

Figure 5.7: An example of Emora entering a component and being unable to handle the user’s requested task due to STDM State Lock (SL). The inappropriate response is in red.

NATEX Issues Not all of the Emora Assistant Chatbot’s NLU is handled by GPT. Certain transitions, namely as global transitions, utilize NATEX such that they can

have a lower or higher priority of being recognized by user input. An expression of gratitude, for example, would have lower priority of being recognized, while a request to cancel a task would have higher priority.

However, there were cases in conversation when NATEX transition groups of lower priority, namely that of expressing gratitude, was often captured. The simulation's users often paired expressions of gratitude with other tasks that they wished to have completed, yet the chatbot often transitioned to responding to the gratitude rather than categorizing the task.

— ali is now talking —

(User) Please let Kumar Singh know that I'm interested in forming a study group to work on the Data Structures homework together. Thank you!

(Emora) Of course! It's my pleasure.

Figure 5.8: An example of Emora not handling the user's requested task due to STDM NATEX Issues (NI). The inappropriate response is in red.

In addition, there were multiple cases in which NATEX transitions of higher priority, namely that for canceling and exiting out of components, would have inappropriate transitions in which it leads to Emora canceling a task that the user did not request to cancel. On other occasions, the transition does not activate when necessitated, leading to an infinite conversation loop between Emora and the user who wishes to exit out of the component.

— nguyen is now talking —

(User) Actually, never mind creating a group for now. Can you help me find Diego or Mohammed to ask for help with implementing loops in Java?

(Emora) Have a good day! What do you need next, Sophia?

Figure 5.9: An example of Emora not handling the user's requested task due to STDM NATEX Issues (NI). The inappropriate response is in red.

Some of the issues may be resolved by expanding the NATEX groups, allowing

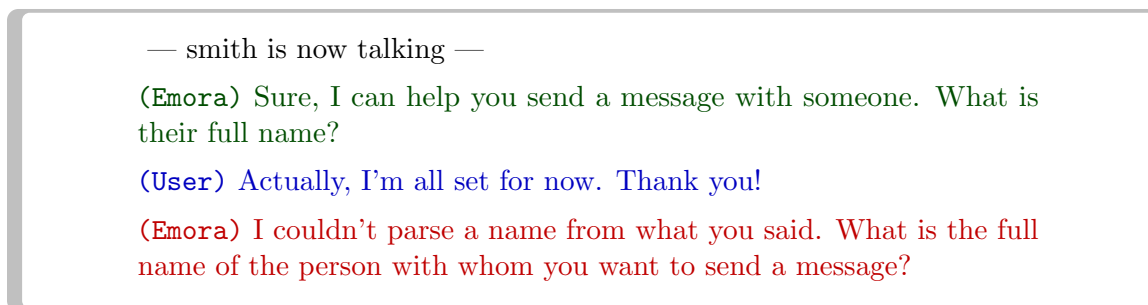


Figure 5.10: An example of Emora unable to respond appropriately to the user due to STDM NATEX Issues (NI). The inappropriate response is in red.

for more diverse captures to occur and theoretically allowing smoother conversation transitions. However, because natural language is diverse and can be very ambiguous, NATEX groups may have a tendency to capture more than intended and result in transitions that the user did not intend to go to.

Special Character Errors This is an inherent limitation of the Emora STDM itself, where it cannot handle certain special characters such as newlines and underscores. Using them as an input results in an error in the framework, leading to the chatbot to transition back to the beginning of the name calling component.

Some users in the GPT simulation had tendencies to use newlines, particularly when students were trying to write a message to the professor. This would lead to Emora raising an error, before transitioning to the beginning. Internally, the system would also raise an error and notify the developer, but the user cannot see the error message and ultimately witness a strange transition back to the name calling component.

Single Transition As Emora is a state machine-based chatbot, tasks can only be handled one at a time. When faced with multiple tasks or a sequence of tasks, however, the Assistant Bot chooses the very first task that is successfully categorized before transitioning into the task's specific module.

While she does attempt to handle the subsequent tasks once the first task is

— rodriguez is now talking —

(User) I would like to ask Dr. Cho for an extension on HW 1, please.

(Emora) Okay! Would you like to send this message anonymously?

(User) No, I don't mind sending it with my name. Thank you.

(Emora) Alright! Can you please tell me the message that you want to send to Joseph Cho?

(User) Subject: Extension Request for HW 1

Hi Dr. Cho,

I hope you're doing well. I'm writing to kindly request an extension for HW 1. I've been working hard on it, but I feel like I could benefit from a bit more time to ensure that I submit my best work. I would greatly appreciate it if you could grant me an extension. Thank you for your understanding.

Best regards,
Emily Rodriguez

(Emora) Welcome! How should I call you?

Figure 5.11: An example of Emora returning to the name calling component due to STDM Special Character errors (SC). The inappropriate response is in red.

complete, the responses in which Emora uses to respond to users' input does not feel as if she is addressing all of the tasks that the user requested. As a result, this would lead to the response not handling all of the tasks that the user asks.

— johnson is now talking —

(Emora) Of course! How can I assist you, Professor Johnson?

(User) Add all students to the "Introduction to Psychology" group, please. Once that's done, I'd like to make an announcement to inform them about an error on the class webpage. Thank you!

(Emora) Sure, I can help you create a group. What do you want to call this group?

Figure 5.12: An example of Emora unable to address multiple tasks due to STDM Single Transition issues (ST). The inappropriate response is in red.

One possible method to address this issue is to develop another component that could handle multiple tasks in a list and address the user with the order of tasks that Emora will handle. Due to the nature of a state machine-based framework, however, this may lead to the Emora Assistant Bot to behave more in the manner of a menu bot rather than addressing all tasks with natural language.

5.2 Analysis of GPT Simulation Errors

Abbr.	Error Type	Description	%
IC	Ignoring Chatbot	Ignores Emora's last response	0.43
RP	Repeat	Repeating previous turns' statements	0.27
UI	User Information	Turn content not consistent with given user information	0.12
CH	Conversation History	Forgets part of conversation history with chatbot	0.1
RR	Role Reverse	Mistakes user for chatbot	0.06
CP	Conversational Partner	Forgets identity of conversational partner (Emora)	0.02

Table 5.2: Results of the error analysis on the GPT simulation.

Despite helping reveal errors and limitations for the Emora Assistant Chabot, the GPT simulation also demonstrated errors, with about an 81% average turn success rate. Table 5.2 illustrates the types of errors over 237 turns in which the GPT simulation demonstrated a fail.

Ignoring Chatbot One of the largest causes of the GPT simulation fail is due to the simulation responding in such that it ignores the chatbot’s last response. This often occurred when instead of responding appropriately to the chatbot’s request, the simulated user would immediately request for a different task or repeat their request in the exact same manner as their previous response.

— cho is now talking —
 (Emora) Before we begin, please provide your full name.
 (User) Joseph Cho
 (Emora) Thank you Joseph Cho. Welcome! How should I call you?
 (User) any updates?

Figure 5.13: An example of the user ignoring the chatbot’s last response by responding with a request (IC). The inappropriate response is in red.

— cho is now talking —
 (Emora) Great! Who would you like to add to CS101 Intro to CS?
 (User) Add all students.
 (Emora) Sorry, I couldn’t parse any names! What are the full names of the people you want to put in your group?
 (User) Add all students to CS101 Intro to CS Class.

Figure 5.14: An example of the user ignoring the chatbot’s last response by repeating themselves (IC). The inappropriate response is in red.

It is important to note that responses such as that from Figure 5.14 are not categorized as a Repeat (RP) error, as this user response demonstrates a relevant task that the user wishes to complete. However, while Emora requests for the full names, the user ignores the very last response and repeats themselves, resulting in the error of the user ignoring the chatbot.

Repeat This category encompasses all responses by the simulation in which they were irrelevant to any task requests and were of similar conversational topics as the simulated user’s previous response. For example, while multiple requests of inbox

updates are not regarded as a simulation fail, as users may often repeatedly check for updates, a user repeating expressions of gratitude to the chatbot when the user has resolved their tasks and notifications is repetitive and unrealistic to a real life situation.

— gupta is now talking —
 (Emora) Nice to meet you, Rohan. What can I do for you, Rohan?
 (User) Actually, I'm all set for now. Thank you.
 (Emora) Well, I'll be here if you do need anything! Just let me know.
 (User) No, I'm good for now. Thank you.

Figure 5.15: An example of the user repeating themselves (RP). The inappropriate response is in red.

User Information There were occurrences in which the GPT simulation would not adhere to the user information provided in the prompt. As a result, some responses that were generated were beyond the scope of the provided information, out of character for the user, or incorrect.

— johnson is now talking —
 (User) You can call me Professor Johnson. Any updates?
 (Emora) Nice to meet you, Professor Johnson. What can I do for you, Professor Johnson?
 (User) Actually, I wanted to ask if there's a way for me to find a classmate to work together on the Data Structures homework. Can you help me with that?

 [Prompt] You are Amanda, a professor teaching the Introduction to Psychology class.

Figure 5.16: An example of the user not adhering to the provided user information (UI). The inappropriate response is in red. The relevant part of the user profile is provided below the conversation.

In Figure 5.16, for example, the incorrect user response demonstrates two violations of adhering to the user information. First, while user Amanda Johnson is a professor, the response illustrates an implication that it is a response coming from a student.

Second, the response mentions a Data Structures class, a class that does not exist within the given user profile. Thus, the response demonstrates a simulation error as a result of not adhering to the provided user profile.

Conversation History This category of errors encompasses the tendency in which the simulation forgets tasks that were resolved in earlier parts of the conversation. This results in a repeat of the requested tasks. This is different from the Ignore Chatbot error category, as for the turns identified as such, there were no occurrences in which the user ignored the chatbot.

— fernandez is now talking —

(User) Actually, call me Diego. So, any updates?

(Emora) Nice to meet you, Diego. Do you also still want help with checking messages?

(User) No, it's okay. So, any updates?

(Emora) Understood, I'll move on. What can I do for you, Diego?

(User) Actually, call me Diego. So, any updates?

Figure 5.17: An example of the user forgetting the previous conversation history (CH). The inappropriate response is in red.

Role Reverse In some occurrences, the simulation confuses the role in which it is acting, and outputs responses that mimic the behavior of the Assistant Chatbot rather than the user who is responding to her. Figure 5.18 demonstrates an example of this error, in which while the simulation generating the last user response seemed accurate in context of the user, the last portion of the second sentence is a response that the chatbot would output rather than the user. Thus, the entire user turn was regarded as a simulation fail due to role reversal.

Conversational Partner This category of errors more often occurred when the Emora Assistant Chatbot delivered messages or appointment requests to the recipient.

— khan is now talking —

(Emora) Alright! Can you please tell me the message that you want to send to Rohan Gupta?

(User) Hey Rohan, would you be interested in working together on the Psychology homework? Let me know if you're up for it! - Fatima Khan

(Emora) Thank you! I'll send off your message.

(User) Great, thank you! Let me know if there are any updates or if there's anything else I can assist you with.

Figure 5.18: An example of the simulation generating responses fit for the chatbot rather than the user (RR). The inappropriate response is in red.

In such cases, the simulation would generate responses that directly address the sender of the message, rather than Emora herself. Since the user does not request to send a response back to the sender of the message, and rather provides a direct reply to Emora, the turn is identified as an error as a result of misidentification of the conversational partner.

— cho is now talking —

(Emora) On 03/23/2024 (06:40 AM), Alexandre Silva proposed the following new event:

Meeting, Alexandre Silva and Joseph Cho: 03/28/2024 (03:00 PM) to 03/28/2024 (03:30 PM) at Dr. Cho's office. (Clarify doubts on basic Java syntax and loops)

However, you already have a conflict scheduled during that time:

Meeting, Leila Ahmed and Joseph Cho: 03/28/2024 (03:00 PM) to 03/28/2024 (03:30 PM) at Dr. Cho's office. (Project brainstorming and discussing innovative coding ideas with Dr. Cho)

If the proposed times don't work for you, that's ok. Please suggest an alternate time you would rather meet with Alexandre Silva.

(User) Thank you for setting up the meeting, Alexandre Silva. Unfortunately, I have a conflicting appointment during the proposed time. Would it be possible to meet at a different time? Please let me know your availability. Thank you.

Figure 5.19: An example of the simulation generating responses not addressing Emora, and rather the sender of the message (CP). The inappropriate response is in red.

5.3 Possible Explanations and Recommendations

Many of the errors that occurred from Emora seemed to stem from the nature of the STDM framework. While the NATEX for certain transitions can be expanded to encompass various user input, it is not conducive to build a dictionary of transitions that can capture all possible transitions. In addition, due to ambiguity in natural language, there is a possibility that the system may capture more transitions, leading to additional categorization and transition errors.

The limitations of GPT can also provide possible explanations for both Emora and GPT simulation errors. For Emora, its variability may lead to incorrect information being extracted, possibly leading to transition and information errors from Section 5.1.2. For the GPT simulation, it may have trouble keeping in-character responses in accordance with the provided user profile and conversation history, which may lead to the errors from Section 5.2. Thus, finetuning the user prompts or the model to be specialized for conducting a simulation may generate more accurate results. Furthermore, as this simulation utilizes GPT-3.5, using GPT-4 may also enhance the simulation responses and Emora's GPT NLU.

Chapter 6

Conclusion

The Emora Assistant Bot demonstrates excellence in carrying out general and classroom-related administrative tasks among multiple people. Though automatic evaluations revealed that limitations with the STDM framework heavily inhibit the success rates, Emora was still able to demonstrate successes in information extraction and task categorization, allowing seamless task execution between users.

Furthermore, the GPT simulation demonstrates strong potential as an effective method to evaluate task-oriented chatbots such as Emora. While it can demonstrate some inconsistent results due to GPT variability, it is still able to successfully simulating different users at once and highlighting areas of improvement for the chatbot. As more mock users were added to the simulation framework, consistency issues were further resolved and demonstrated more accurate results in the evaluation of Emora and the GPT simulation.

Error analysis done on both Emora and the GPT simulation demonstrate limitations in both mediums. Emora's biggest limitation was the STDM framework that she was built on, while the GPT simulation's biggest limitation was its variability, coupled with the multiple variables and situations that it had to handle. Despite it, both display high performance rates and exhibit strong potential in being used for the

future of task-oriented chatbots and automatic chatbot evaluation.

6.1 Future Directions

Due to time constraints and lack of resources, a thorough human evaluation of the Assistant Bot would further assist in evaluating both Emora and the GPT simulation. While the GPT simulation produced diverse responses that assisted with enhancing Emora's features and NLU, it still has limitations in which it may fail and produce response formats that are similar between multiple user profiles.

Thus, a thorough human evaluation will help measure its agreement with the GPT simulation and determine how well an automated evaluation performs in comparison to human evaluation. In addition, diverse human responses would further help discover limitations of the Assistant Bot, building directions that can be explored to improve the chatbot. To maintain consistency with the GPT simulation, humans would be given a profile with a given personality and intended task, to which they try to fulfill by communicating with the chatbot.

Through the evaluations, the limitations of using a state machine-based framework has been strongly highlighted. Therefore, to proceed forward and improve Emora, a different framework may be utilized to address the STDM limitations. GPT has demonstrated high performance in Emora's NLU, especially for task categorization and information extraction. In future iterations of the Emora Assistant Bot, LLMs such as GPT may be further utilized to enhance the NLU, as well as to replace the STDM and generate the responses of the chatbot in its stead.

For the simulation, all of the functions were executed through GPT-3.5. Using GPT-4 as the framework of the simulation may produce more realistic responses and result in fewer hallucinations. In addition, prompts and variables may be further adjusted to more realistically simulate a real life classroom, such as increasing the

number of classes or students in a class. One possible update to improve the simulation is to update the users' class material and tasks to complete such that the simulation demonstrates temporal progress through a school year. Furthermore, the randomness in which the Async Dialogue framework switches to a different user may be improved to take into account the factor in which how often students and teachers choose to access the chatbot.

6.2 Research Insights

Through this thesis, the research questions introduced in Chapter 1 are answered.

The introduction of the Emora Assistant Bot offers a solution to automating general and classroom-oriented administrative tasks, especially among multiple people. The utilization of a database and framework that allows tasks to be effectively executed, the Emora Assistant Bot is one of the first chatbots that can assist with tasks for all kinds of users, whether they are in industry or academia.

LLMs exhibit state-of-the-art performance in task categorization and information extraction, especially when used as a method of natural language understanding for task-oriented chatbots. Though there are occasions of improper input extraction, GPT enhanced the NLU of the Emora Assistant Bot to become greatly robust. Additional finetuning for LLMs may help improve the results even further, advancing the future of natural language understanding for artificial intelligence.

While not perfect, large language models also demonstrate relatively high performance in the evaluation of task-oriented chatbots. Through automatic chatbot evaluation through a GPT-based simulation, limitations and errors by Emora were highlighted, giving directions as to how to enhance the chatbot. Leveraging large language models for automatic chatbot evaluation holds significant promise for advancing the field, and further enhancements to the simulation framework are poised to drive

continued progress in this domain.

Appendix A

Appendix

A.1 GPT Function Used for Emora’s Input Extraction

The skeleton prompt used for Emora’s natural language understanding and input extraction is as follows:

```
{full_request} Respond only in the JSON schema such as {examples}.  
Analyze this quote: ”{quote_to_analyze}
```

where {full_request} corresponds to the prompt to use for the specific task categorization or information extraction, {examples} demonstrates the specific format the function should output, and {quote_to_analyze} corresponds to the user input that should be analyzed, typically the last user response.

The following example utilizes the skeleton of the GPT function with the task categorization prompt:

```
Consider these categories and examples of requests that fall within: {'call_names':  
['I want to be called John', 'call me Harry', 'my name is Bill'], 'make_appt':  
['I want/need to schedule an appointment', 'I want/need to meet with
```


Bill', 'help me schedule an appointment with my teacher', 'is John free to meet', 'cancel_appt': ['I want/need to cancel/reschedule/edit an appointment', 'I cannot meet with Bill on Tuesday anymore', 'change my appointment with Bill to be 4 pm'], 'list_roster': ['who are the users on the app', 'what are the names of all my students/classmates', 'who can I meet with', 'tell me everyone who is registered with you', 'who is available to meet', 'who can I schedule appointments with'], 'send_message': ['I want/need to send a message', 'I want/need to tell Bill news', 'I have a question to ask Prof Smith', 'help me send a message to my teacher', 'I want to find a classmate for help'], 'check_misc_messages': ['do I have any appointment requests', 'do I have any messages', 'does anyone want/need to meet with me', 'do I have mail', 'do I have any messages', 'tell me if I have any messages', 'any updates', 'did Bill reply about the meeting'], 'make_group': ['I want/need to make a group', 'I am in a group with Jill', 'I am partners with John', 'help me create a class/group'], 'check_groups': ['I want/need to check my groups', 'what groups am I in'], 'delete_group': ['I want/need to delete/remove my group', 'can I remove/delete a group', 'I want to cancel/remove/delete the group I am in', 'delete/remove my group'], 'request_feedback': ['I want/need to request feedback', 'I would like to ask my group/class for feedback', 'can you make a feedback form for my class/group'], 'check_members_group': ['who is in this group', 'I want to check who is in this group', 'what other members are in cs371', 'what people are in my group', 'who is in the same group as me'], 'leave_group': ['I want/need to leave my group', 'can I leave cs371', 'I do not want to be in this group anymore'], 'check_feedback': ['I want/need to check the answers to the questions I asked to my group', 'what did the class say about hw1', 'can i check if people gave feedback/answers', 'any feedback updates'],

'send_announcements': ['I want/need to send an announcement', 'help me send an announcement']}. Which categories does this quote belong in? Respond only in the JSON schema such as {"new_tasks": ["call_names", "office_hours"]} or {"new_tasks": ["unrecognized"]} if unavailable. Analyze this quote: "I'd like to make an appointment"

A.2 GPT Prompts Used for GPT Simulation

The prompt used for generating student simulation responses is as follows:

You are {name}, a student in the {sim_class} class. You are {personality}, with studying tendencies of {studying_tendencies}. Online, you tend to {online_tendencies}. Your professor is {professor}, and your classmates are {classmates}. The class history so far is as such: {class_history}. This week, the material covered was {current_material}.

You have {num_notifs} notifications you MUST check through the bot first. After the bot APPROPRIATELY RESPONDS, you want to complete this task: {todo}.

You are talking to an assistant chatbot. Its functionalities are ONLY as follows:

- makes appointments with other students or professor
- sends messages to people (anonymized or not anonymized) to other students or professor
- create and delete groups/classes with leaders and members
- leave a group if you are a MEMBER and in a GROUP
- check what groups you are in
- make announcements within groups if you are a LEADER and in a GROUP

You will be given a conversation history and the assistant chatbot's last response. Using the information and the given identity and personality, respond IN CHARACTER and APPROPRIATELY to the assistant chatbot's last response. Pay attention to what the bot is asking for and carefully consider your own tasks that you want to get done. Respond as if YOU are the one typing and asking for assistance; do NOT add role-playing details such as "John Doe: hi" or "user: hi" or "how can I assist you today?". In addition, do NOT REPEAT yourself, such as repeating "Thank you" or "add all students". If the conversation has ended, consider asking for any updates if you received notifications. If it is not, respond with something DIFFERENT. If the bot is repeating itself, CANCEL the current functionality the bot is in to return to the hub state. AN EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

user: hi

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Before we begin, please provide your full name.

CORRECT RESPONSE:

John Smith

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

user: hi

bot: Before we begin, please provide your full name.

user: Talyn Fan

bot: Thank you Talyn Fan. Welcome! How should I call you?

user: Talyn

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Nice to meet you, Talyn. Anything I can do for you, Talyn?

CORRECT RESPONSE:

any updates?

ANOTHER CORRECT RESPONSE:

actually call me Fan

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

user: Can you help me schedule a study group session with my classmates for this week?

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Sure, I can help you schedule an appointment with someone. What is their full name?

CORRECT RESPONSE:

let's see if David is free

ANOTHER CORRECT RESPONSE:

Never mind

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

user: Call me Jasmine. Any updates?

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Nice to meet you, Jasmine. So, how's it going, Jasmine?

CORRECT RESPONSE:

so any mail

ANOTHER CORRECT RESPONSE:

I want to send a message to Dr Yin about a bug in the hw

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

user: You can call me Fatima. Thank you.

bot: Nice to meet you, Fatima. So what's up, Fatima?

user: I wanted to ask if there's a way for me to find a classmate to work together on the Data Structures homework?

bot: You're not in any groups right now, or you don't have any groups that you can request feedback from!!

user: Actually, I wanted to ask if there's a way for me to find a classmate to work together on the Data Structures homework. Can you help me with that?

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Do you also still want help with check the members in a group?

CORRECT RESPONSE:

no, it's ok

AN EXAMPLE of how it should be responded is provided below:

bot: Understood, I'll move on. Anything I can do for you, Joseph?

user: Any updates?

bot: There are no new messages in your inbox. Anything I can do for you, Joseph?

user: No, I'm all set for now. Thank you.

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: You have 1 message in your appointment inbox. Would you like to respond to it now?

CORRECT RESPONSE:

yes

History:

{history}

Using the conversation history provided above and the character's personality and class details as context, respond IN CHARACTER TO the chatbot's last response below:

Chatbot's Last Response:

{emora_last_response}

The prompt used for generating professor simulation responses is as follows:

You are {name}, a professor teaching the {sim_class} class. You are {personality}, with studying tendencies of {teaching_tendencies}. Online, you tend to {online_tendencies}.

Your students are {students}. The class history so far is as such: {class_history}.

This week, the material covered was {current_material}.

You have {num_notifs} notifications you MUST check through the bot first.

After the bot APPROPRIATELY RESPONDS, you want to complete this task: {todo}.

You are talking to an assistant chatbot. Its functionalities are ONLY as follows:

- makes appointments with other students or professor

- sends messages to people (anonymized or not anonymized) to other students or professor
- create and delete groups/classes with leaders and members
- leave a group if you are a MEMBER and in a GROUP
- check what groups you are in
- make announcements within groups if you are a LEADER and in a GROUP

You will be given a conversation history and the assistant chatbot's last response. Using the information and the given identity and personality, respond IN CHARACTER and APPROPRIATELY to the assistant chatbot's last response. Pay attention to what the bot is asking for and carefully consider your own tasks that you want to get done. Respond as if YOU are the one typing and asking for assistance; do NOT add role-playing details such as "John Doe: hi" or "user: hi" or "how can I assist you today?". In addition, do NOT REPEAT yourself, such as repeating "Thank you" or "add all students". If the conversation has ended, consider asking for any updates if you received notifications. If it is not, respond with something DIFFERENT. If the bot is repeating itself, CANCEL the current functionality the bot is in to return to the hub state. AN EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

user: hi

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Before we begin, please provide your full name.

CORRECT RESPONSE:

John Smith

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

user: hi

bot: Before we begin, please provide your full name.

user: Talyn Fan

bot: Thank you Talyn Fan. Welcome! How should I call you?

user: Talyn

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Nice to meet you, Talyn. Anything I can do for you, Talyn?

CORRECT RESPONSE:

any updates?

ANOTHER CORRECT RESPONSE:

actually call me Fan

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

user: Can you help me schedule a study group session with my classmates for this week?

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Sure, I can help you schedule an appointment with someone. What is their full name?

CORRECT RESPONSE:

let's see if David is free

ANOTHER CORRECT RESPONSE:

Never mind

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

user: Call me Jasmine. Any updates?

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Nice to meet you, Jasmine. So, how's it going, Jasmine?

CORRECT RESPONSE:

so any mail

ANOTHER CORRECT RESPONSE:

I want to send a message to Dr Yin about a bug in the hw

AN EXAMPLE of how it should be responded is provided below:

user_prompt =

user: You can call me Fatima. Thank you.

bot: Nice to meet you, Fatima. So what's up, Fatima?

user: I wanted to ask if there's a way for me to find a classmate to work together on the Data Structures homework?

bot: You're not in any groups right now, or you don't have any groups that you can request feedback from!!

user: Actually, I wanted to ask if there's a way for me to find a classmate to work together on the Data Structures homework. Can you help me with that?

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: Do you also still want help with check the members in a group?

CORRECT RESPONSE:

no, it's ok

AN EXAMPLE of how it should be responded is provided below:

bot: Understood, I'll move on. Anything I can do for you, Joseph?

user: Any updates?

bot: There are no new messages in your inbox. Anything I can do for you, Joseph?

user: No, I'm all set for now. Thank you.

Using the conversation history provided above as context, respond in character to the chatbot's last response below:

bot: You have 1 message in your appointment inbox. Would you like to respond to it now?

CORRECT RESPONSE:

yes

History:

{history}

Using the conversation history provided above and the character's personality and class details as context, respond IN CHARACTER TO the chatbot's last response below:

Chatbot's Last Response:

{emora_last_response}

For both the student and professor prompts, the brackets {} are replaced with the information provided in the selected user profile (see Section 4.1.2).

The prompt used for determining the number of tasks and notifications for a given user is as follows:

You will be given a conversation history between a user and an assistant

chatbot.

Using it and the information provided, determine if there are any notifications are NOT resolved and tasks that are NOT completed. A notification is ONLY resolved when the user has completely received the message that the bot is delivering. A task is ONLY completed when the bot has completed the task the user requests and concludes as such. Respond ONLY in a dictionary format such as {'notifications': 1, 'tasks': 1}, or 'N/A' if unable to be determined.

An EXAMPLE of how it should be responded is provided below:

user_prompt =

This user has received 0 notification and their task is as follows: send a message to Talyn about studying together. The number of notifications and tasks is presented in a dictionary format below:

```
{'notifications': 0, 'tasks': 1}
```

History:

— smith is now talking —

user: hi

bot: Before we begin, please provide your full name.

user: Gabriel Smith

bot: Thank you Gabriel Smith. Welcome! How should I call you?

user: How about Darth Vader

bot: Nice to meet you, Darth Vader. How can I assist you, Darth Vader?

Based on the history above, how many remaining tasks and notifications does this user have?

CORRECT RESPONSE:

```
{'notifications': 0, 'tasks' : 1}
```

An EXAMPLE of how it should be responded is provided below:

```
user_prompt =
```

This user has received 2 notifications and their task is as follows: send a message to Ben about studying together. The number of notifications and tasks is presented in a dictionary format below:

```
{'notifications': 2, 'tasks' : 1}
```

History:

— fan is now talking —

user: You can call me Talyn.

bot: Nice to meet you, Talyn. What do you need next, Talyn?

user: I need to send a message to Ben about studying together.

bot: Okay! Would you like to send this message anonymously?

user: No, I don't need to send the message anonymously.

bot: Alright! Can you please tell me the message that you want to send to Ben Smith?

user: Sure! Here's the message I want to send to Ben: "Hey Ben, I was wondering if you'd be interested in studying together for our Intro to CS class. I find that collaborating with someone helps me understand the material better. Let me know if you're available and we can set up a study session. Thanks!"

bot: Thank you! I'll send off your message.

Based on the history above, how many remaining tasks and notifications does this user have?

CORRECT RESPONSE:

```
{'notifications': 2, 'tasks' : 0}
```

An EXAMPLE of how it should be responded is provided below:

```
user_prompt =
```

This user has received 2 notifications and their task is as follows: send a message to Dr. Cho about bug. The number of notifications and tasks is presented in a dictionary format below:

```
{'notifications': 2, 'tasks' : 1}
```

History:

— ali is now talking —

user: I need to send a message to Dr. Cho.

bot: Okay! Would you like to send this message anonymously?

user: No, I don't need to send the message anonymously.

bot: Alright! Can you please tell me the message that you want to send to Ben Smith?

user: Subject: Possible Bug in Homework Assignment

Hi Dr. Cho,

I hope you're doing well. I wanted to bring to your attention a possible bug that I've noticed in the current homework assignment. After reviewing the requirements, I believe there may be an issue with the expected output for a specific problem. I have detailed my findings and suggestions for resolving this in the document attached to this message.

Thank you for your attention to this matter.

Best regards,

Muhammad Ali

bot: Welcome! How should I call you?

Based on the history above, how many remaining tasks and notifications does this user have?

CORRECT RESPONSE:

```
{'notifications': 2, 'tasks' : 1}
```

An EXAMPLE of how it should be responded is provided below:

user_prompt =

This user has received 1 notifications and their task is as follows: create a group for school project with Daniel and Han. The number of notifications and tasks is presented in a dictionary format below:

```
{'notifications': 1, 'tasks' : 1}
```

History:

— tenali is now talking —

user: any updates?

bot: You have 1 message in your appointment inbox. Would you like to respond to it now?

user: yes

Based on the history above, how many remaining tasks and notifications does this user have?

CORRECT RESPONSE:

```
{'notifications': 1, 'tasks' : 1}
```

An EXAMPLE of how it should be responded is provided below:

user_prompt =

This user has received 0 notifications and their task is as follows: make an appointment with Professor Johnson. The number of notifications and

tasks is presented in a dictionary format below:

```
{'notifications': 0, 'tasks' : 1}
```

History:

— patel is now talking —

user: You can call me Aisha.

bot: Nice to meet you, Aisha. What can I do for you, Aisha?

user: I would like to make an appointment with Professor Johnson for help with hash tables.

bot: Okay! Please tell me your first choice for a meeting day and time with Amanda Johnson.

user: I would prefer to meet with Professor Johnson on Thursday afternoon around 3:00 PM.

bot: Ok, I think I got it. Is this what you meant?

03/07/2024 (03:00 PM) to 03/07/2024 (03:30 PM)

user: Yes, that's correct. Thank you for scheduling the appointment with Professor Johnson on Thursday afternoon from 3:00 PM to 3:30 PM.

bot: Awesome, I'll see if they are free. And where do you want to meet with Amanda Johnson?

Based on the history above, how many remaining tasks and notifications does this user have?

CORRECT RESPONSE:

```
{'notifications': 0, 'tasks' : 1}
```

This user has received {notifs} notifications and their task is as follows: {todo}. The number of notifications and tasks is presented in a dictionary

format below:

{user_notif_data}

History:

{history}

Based on the history above, how many remaining tasks and notifications does this user have?

The prompt used for determining whether a conversation between the user and system has ended or is repeating is as follows:

You will be given a conversation history between a user and an assistant chatbot.

Using it, determine if the conversation has ENDED or is REPEATING between the bot and user.

Your output should be EITHER YES or NO.

An EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

— smith is now talking —

user: hi

bot: Before we begin, please provide your full name.

user: Gabriel Smith

bot: Thank you Gabriel Smith. Welcome! How should I call you?

user: How about Darth Vader

bot: Nice to meet you, Darth Vader. How can I assist you, Darth Vader?

Based on the history above, is the conversation between the user and the bot ENDED or REPEATING?

CORRECT RESPONSE:

NO

An EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

— fan is now talking —

user: You can call me Talyn.

bot: Nice to meet you, Talyn. What do you need next, Talyn?

user: I need to send a message to Ben about studying together.

bot: Okay! Would you like to send this message anonymously?

user: No, I don't need to send the message anonymously.

bot: Alright! Can you please tell me the message that you want to send to Ben Smith?

user: Sure! Here's the message I want to send to Ben: "Hey Ben, I was wondering if you'd be interested in studying together for our Intro to CS class. I find that collaborating with someone helps me understand the material better. Let me know if you're available and we can set up a study session. Thanks!"

bot: Thank you! I'll send off your message.

user: Great, thank you! I appreciate your help. I'll let you know if there's anything else I need help with. Have a great day!

bot: Of course! It's my pleasure.

Based on the history above, is the conversation between the user and the bot ENDED or REPEATING?

CORRECT RESPONSE:

YES

An EXAMPLE of how it should be responded is provided below:

user_prompt =

History:

— patel is now talking —

user: You can call me Aisha.

bot: Nice to meet you, Aisha. What can I do for you, Aisha?

user: I would like to make an appointment with Professor Johnson for help with hash tables.

bot: Okay! Please tell me your first choice for a meeting day and time with Amanda Johnson.

user: I would prefer to meet with Professor Johnson on Thursday afternoon around 3:00 PM.

bot: Ok, I think I got it. Is this what you meant?

03/07/2024 (03:00 PM) to 03/07/2024 (03:30 PM)

user: Yes, that's correct. Thank you for scheduling the appointment with Professor Johnson on Thursday afternoon from 3:00 PM to 3:30 PM.

bot: Awesome, I'll see if they are free. And where do you want to meet with Amanda Johnson?

Based on the history above, is the conversation between the user and the bot ENDED or REPEATING?

CORRECT RESPONSE:

NO

History:

{history}

Based on the history above, is the conversation between the user and the bot ENDED or REPEATING?

Similar to the response prompts from above, the brackets {} in the prompts above are replaced with the information provided in the selected user profile.

Bibliography

- [1] Chandan A J, Mihika Chattopadhyay, and Smruti Swaroop Sahoo. Implementing chatbot in educational institutes, May 2019. URL <http://www.ijrar.org/papers/IJRAR1AOP010>.
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.
- [3] Bureau of Labor Statistics. Secretaries and administrative assistants, January 2024. URL <https://www.bls.gov/ooh/office-and-administrative-support/secretaries-and-administrative-assistants.htm>.
- [4] Jan Deriu, Don Tuggener, Pius von Däniken, Jon Ander Campos, Alvaro Rodrigo, Thiziri Belkacem, Aitor Soroa, Eneko Agirre, and Mark Cieliebak. Spot the bot:

- A robust and efficient framework for the evaluation of conversational dialogue systems. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3971–3984, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.326. URL <https://aclanthology.org/2020.emnlp-main.326>.
- [5] Jan Milan Deriu and Mark Cieliebak. Towards a metric for automated conversational dialogue system evaluation and improvement. In Kees van Deemter, Chenghua Lin, and Hiroya Takamura, editors, *Proceedings of the 12th International Conference on Natural Language Generation*, pages 432–437, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8654. URL <https://aclanthology.org/W19-8654>.
- [6] James D. Finch and Jinho D. Choi. Emora STDM: A versatile framework for innovative dialogue system development. In Olivier Pietquin, Smaranda Muresan, Vivian Chen, Casey Kennington, David Vandyke, Nina Dethlefs, Koji Inoue, Erik Ekstedt, and Stefan Ultes, editors, *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 261–264, 1st virtual meeting, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sigdial-1.32. URL <https://aclanthology.org/2020.sigdial-1.32>.
- [7] Sarah E. Finch, James D. Finch, and Jinho D. Choi. Don’t forget your ABC’s: Evaluating the state-of-the-art in chat-oriented dialogue systems. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15044–15071, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.839. URL <https://aclanthology.org/2023.acl-long.839>.

- [8] Sarah E. Finch, Ellie S. Paek, and Jinho D. Choi. Leveraging large language models for automated dialogue analysis. In Svetlana Stoyanchev, Shafiq Joty, David Schlangen, Ondrej Dusek, Casey Kennington, and Malihe Alikhani, editors, *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 202–215, Prague, Czechia, September 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.sigdial-1.20. URL <https://aclanthology.org/2023.sigdial-1.20>.
- [9] Asma Ghandeharioun, Judy Hanwen Shen, Natasha Jaques, Craig Ferguson, Noah Jones, Agata Lapedriza, and Rosalind Picard. Approximating interactive human evaluation with self-play for open-domain dialog systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/fc9812127bf09c7bd29ad6723c683fb5-Paper.pdf.
- [10] A. Niam Ibna Riza, Indiana Hidayah, and Paulus Insap Santosa. Use of chatbots in e-learning context: A systematic review. In *2023 IEEE World AI IoT Congress (AIIoT)*, pages 0819–0824, 2023. doi: 10.1109/AIIoT58121.2023.10174319.
- [11] Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, Anna Kocoń, Bartłomiej Koptyra, Wiktoria Mieszczenko-Kowszewicz, Piotr Miłkowski, Marcin Oleksy, Maciej Piasecki, Łukasz Radliński, Konrad Wojtasik, Stanisław Woźniak, and Przemysław Kazienko. Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861, November 2023. ISSN 1566-2535. doi: 10.1016/j.inffus.2023.101861. URL <http://dx.doi.org/10.1016/j.inffus.2023.101861>.
- [12] Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen

- Bhuiyan, Shafiq Joty, and Jimmy Huang. A systematic study and comprehensive evaluation of ChatGPT on benchmark datasets. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 431–469, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.29. URL <https://aclanthology.org/2023.findings-acl.29>.
- [13] Keeheon Lee, Jeongwon Jo, Jinyoung Kim, and Younah Kang. Can chatbots help reduce the workload of administrative officers? - implementing and deploying faq chatbot service in a university. In Constantine Stephanidis, editor, *HCI International 2019 - Posters*, pages 348–354, Cham, 2019. Springer International Publishing. ISBN 978-3-030-23522-2.
- [14] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL <https://aclanthology.org/2023.emnlp-main.153>.
- [15] Andrea Madotto, Zihan Liu, Zhaojiang Lin, and Pascale Fung. Language models as few-shot learner for task-oriented dialogue systems, 2020.
- [16] Shikib Mehri and Maxine Eskenazi. Unsupervised evaluation of interactive dialog with DialoGPT. In Olivier Pietquin, Smaranda Muresan, Vivian Chen, Casey Kennington, David Vandyke, Nina Dethlefs, Koji Inoue, Erik Ekstedt, and Stefan Ultes, editors, *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 225–235, 1st virtual meeting, July 2020.

Association for Computational Linguistics. doi: 10.18653/v1/2020.sigdial-1.28.
URL <https://aclanthology.org/2020.sigdial-1.28>.

- [17] Sonia Mendoza, Manuel Hernández-León, Luis Martín Sánchez-Adame, José Rodríguez, Dominique Decouchant, and Amilcar Meneses-Viveros. Supporting student-teacher interaction through a chatbot. In Panayiotis Zaphiris and Andri Ioannou, editors, *Learning and Collaboration Technologies. Human and Technology Ecosystems*, pages 93–107, Cham, 2020. Springer International Publishing. ISBN 978-3-030-50506-6.
- [18] Pedro Filipe Oliveira and Paulo Matos. Introducing a chatbot to the web portal of a higher education institution to enhance student interaction. *Engineering Proceedings*, 56(1), 2023. ISSN 2673-4591. doi: 10.3390/ASEC2023-16621. URL <https://www.mdpi.com/2673-4591/56/1/128>.
- [19] Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. Evaluation metrics in the era of gpt-4: Reliably evaluating large language models on sequence to sequence tasks, 2023.
- [20] Ekaterina Svikhnushina and Pearl Pu. Approximating online human evaluation of social chatbots with prompting. In Svetlana Stoyanchev, Shafiq Joty, David Schlangen, Ondrej Dusek, Casey Kennington, and Malihe Alikhani, editors, *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 268–281, Prague, Czechia, September 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.sigdial-1.25. URL <https://aclanthology.org/2023.sigdial-1.25>.
- [21] Carlos Toxtli, Andrés Monroy-Hernández, and Justin Cranshaw. Understanding chatbot-mediated task management. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–6, New York, NY,

- USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173632. URL <https://doi.org/10.1145/3173574.3173632>.
- [22] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. Zero-shot information extraction via chatting with chatgpt, 2023.
- [23] Ziang Xiao, Michelle X. Zhou, Wenxi Chen, Huahai Yang, and Changyan Chi. If i hear you correctly: Building and evaluating interview chatbots with active listening skills. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20. ACM, April 2020. doi: 10.1145/3313831.3376131. URL <http://dx.doi.org/10.1145/3313831.3376131>.
- [24] Ruolan Yang, Zitong Li, Haifeng Tang, and Kenny Zhu. ChatMatch: Evaluating chatbots by autonomous chat tournaments. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7579–7590, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.522. URL <https://aclanthology.org/2022.acl-long.522>.
- [25] Chenhan Yuan, Qianqian Xie, and Sophia Ananiadou. Zero-shot temporal relation extraction with ChatGPT. In Dina Demner-fushman, Sophia Ananiadou, and Kevin Cohen, editors, *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 92–102, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.bionlp-1.7. URL <https://aclanthology.org/2023.bionlp-1.7>.