Building Hybrid Task-Oriented Knowledge Support Chatbot

Yi-Qing Lin
University of California, Davis
Davis, California
United States
etlin@ucdavis.edu

ABSTRACT

People frequently encounter challenges in unfamiliar domains, necessitating effective support to fill the knowledge gap. To handle this need, task-oriented chatbot are deployed aiming to assist users in problem-solving while reducing operational costs and human labor requirements. However, interacting with task-oriented chatbot often require users to possess domain knowledge, contradicting their fundamental purpose of providing automated assistance. Additionally, recent integrations with Large Language Model (LLM) have introduced new challenges in user inability to follow instruction and identify response reliability. We present a modularized chatbot that provides accurate responses tailored to user proficiency through a novel architecture combining state-machine-based conversation management and task management. Our approach structures domain knowledge as a comprehensive workflow of goals and operators, enabling more effective automated support. By adopting this knowledge support chatbot, people are expected to have more confidence when operating with the chatbot without the domain knowledge, get more familiar and eventually pick up the skills to operate the system the chatbot is supporting.

1. INTRODUCTION

In daily activities, people frequently encounter situations where they need to perform tasks in unfamiliar domains. For example, performing a task on a new system or troubleshooting unexpected issues. People often seek guidance to bridge their knowledge gaps. Traditional solutions such as searching on search engines for text-based tutorials or forums posts [1, 2, 3, 4] requires ability to articulate queries with precise keywords [2, 5, 6], and seeking assistance from human experts leads to constraints in availability and scalability. The need for effective automated assistance becomes specifically prominent in organizations where support requests cover broad areas and various levels of user proficiency.

The industry has responded to this human-dependent limitation with the wide adoption of automated chatbot, which comes in various forms including menu-based interactive voice response systems and text-based chatbots. They serve as virtual assistants across various domains such as customer service, technical support, and software guidance. Platforms such as IBM Watson and Google DialogFlow was developed to offer user-friendly interfaces to simplify and speed up the development process. While these platforms have significantly shortened production timelines, ensuring a well-designed and cohesive dialogue flow remains essential for maintaining high-quality user interactions.

Despite the widespread deployment, current chatbots are designed to assume users can “ask the right question” using domain-specific terminology or are familiar with the procedural steps involved in a task. This assumption may create a significant usability issue, as users seeking assistance often lack the knowledge necessary to effectively express their needs. For example, a novice photo editor may want to apply an effect to an image, but the user may struggle to locate the relevant button and may not know the precise term to ask the chatbot integrated in the photo editing software.

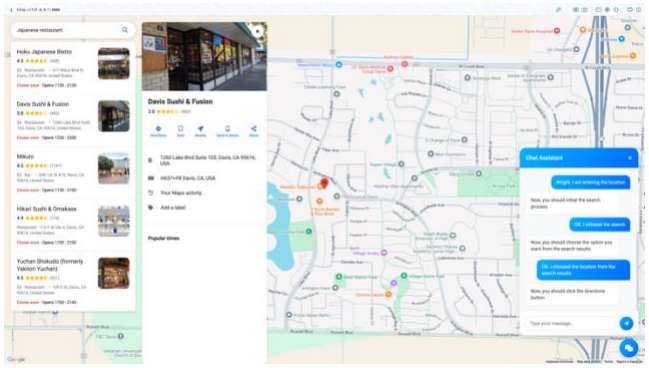


Figure 1. Web application integrated with knowledge support chatbot

Recent advancements in LLMs such as GPT3 [7] and LLaMA [8] have revolutionized dialogue management through their sophisticated natural language understanding and natural language generation capabilities [9]. However, these advanced systems lead to a new set of challenges in user interaction. Users frequently encounter difficulties in formulating effective queries [9], following generated instructions [10], or identifying potential biases in the responses [10, 11]. Moreover, unexpected response could be generated [12] due to the nature of predicting the most probable next word in each sentence. These challenges highlight the importance of developing more robust chatbot that effectively bridge the gap between advanced language processing capabilities and practical user needs.

In this work, we introduce an innovative hybrid chatbot architecture designed to address these challenges by providing accurate responses tailored to user proficiency. The chatbot is composed with three main components: user profile module, natural language processing module, and dialogue engine.

The user profile module contains two units and a user ID. The first unit is the dialogue flow management unit, which is a dual-level state machine that tracks the conversational purpose and handles speech acts. The second unit is the task flow management unit, which is also a dua-level state machine that tracks the

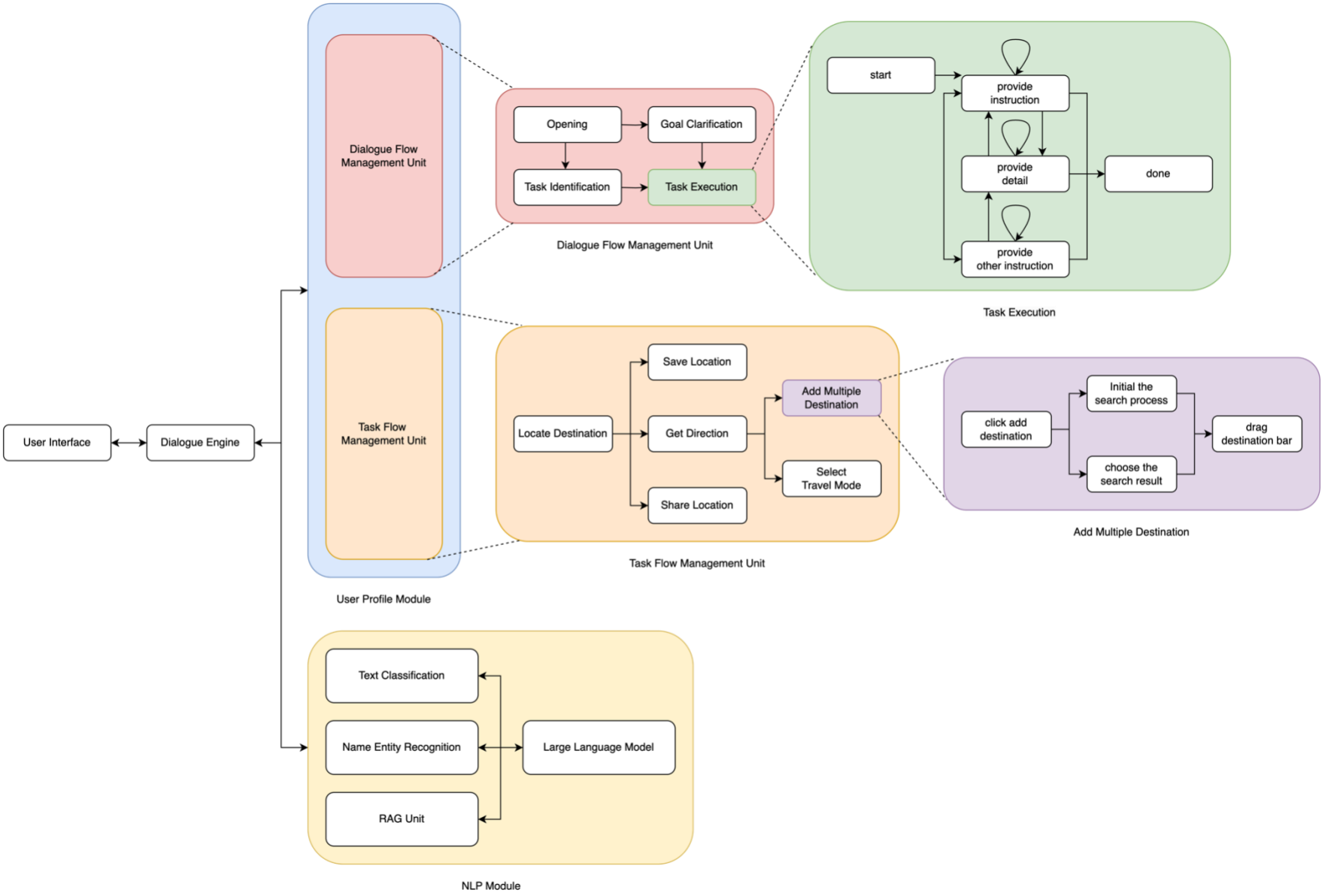


Figure 2. Hybrid Chatbot Architecture

progress of tasks and maintains the user proficiency level according to user actions on the application. The task flow management unit is a comprehensive workflow that contains goal, methods, and operators [13]. Goals are the higher-level objectives such as searching for a location or saving a location to a list. Methods and the middle-level objectives that contains dependencies. For example, user would have to find a place on a map website before saving it to a list. Therefore, any method can be the end of a goal and the whole trajectory is the entire goal to be achieved. Operators are the lower-level, inseparable objectives to achieve methods, such as clicking on a button or typing information into the input field. The chatbot would provide instruction and details according to operators and keep the methods and goal as a blueprint for guiding the user. Therefore, the chatbot evaluates the user proficiency level of each operator to provide tailored guidance, offering sufficient support while avoiding information overload.

The natural language processing module contains three units that are all integrated with LLM for it showing strong capabilities in performing classification jobs [14] and response generation, achieving generalizability. Additionally, we integrated LLM locally to ensure data privacy and system portability. The first and the second units are the text classification unit and the name entity recognition unit that are supported with prompt engineering with LLM. The third unit is the retrieval-augmented generation (RAG) unit that generates response to the user.

Finally, the dialogue engine work as the central conductor that communicates using API to the user interface and other modules. It manages the conversation with the user by coordinating with the dialogue flow management module and task flow management module. This component ensures streamlined information exchange between user and the knowledge support chatbot. This design has significant potential for applications that requires user support and technical assistance that provides accurate response.

2. RELATED WORK

Researchers have extensively studied how to effectively model task-oriented chatbots to enhance user task performance. Paweł et al. [17] emphasize the importance of data-driven approaches in conversational agent development given the challenge of limited training data. In terms of communication style, Cristen et al. [18] showed that utilizing discourse markers and hedges in chatbot responses influences users' perceptions of the system positively. To address language processing challenges, Maite [19] developed a discourse module that converts user inputs into speech acts. This created a standardized language framework for conversational agents and reduced input ambiguity through this translation process. Taking a structural approach, Dan et al. [20] proposed separating dialogue management into domain-dependent and domain-independent components, focusing on task decomposition and conversational strategies. Their research showed that task decomposition improves runtime scalability, while well-designed conversational strategies enhance both chatbot usability and task completion rates.

Researchers have also investigated how user-agent interactions influence conversational dynamics, with particular focus on communication challenges and user engagement. Chi-Hsun et al. [21] conducted a systematic analysis of conversational breakdowns, identifying and categorizing the primary types of non-progressive interactions that impede effective communication. Asbjørn et al. [22] revealed that conversational breakdowns deteriorate overall trust, their impact, which the level

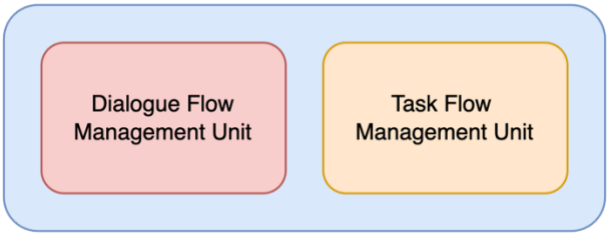


Figure 3. User Profile Unit

varies depending on the breakdown's timing in the interaction sequence, with evidence of trust recovery following successful task completion. Building on this understanding of conversational challenges, Amon et al. [23] developed design guidelines focused on three key objectives: maintaining user engagement in collaborative interactions, guiding users toward optimal collaborative strategies and fostering a sense of progressive collaboration even when encountering difficulties. Their recommendations aim to enhance the resilience and effectiveness of human-agent interactions.

3. HYBRID CHATBOT ARCHITECTURE

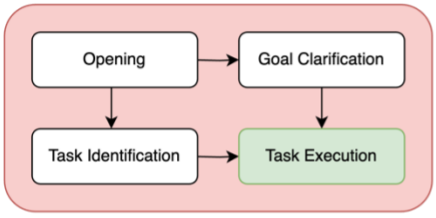
Dialogue system architecture is well-developed field with extensive research and literature. Andrew [15] proposed a generalized Al assistant architecture that balances simplicity with functionality, providing a robust foundation for dialogue systems. The core architecture consists of three components: a user interface, a dialogue engine, and a natural language understanding module. We enhanced this framework by incorporating two additional components: a natural language generation module for producing responses, and a user profile module to store the progress of the conversation and domain knowledge (Figure 2). Additionally, the system dynamically loads the domain knowledge and dialogue flow so that developers can alter these components with ease. In the following sections, we use selected states from the dialogue flow we designed as illustrative examples.

3.1 User Profile

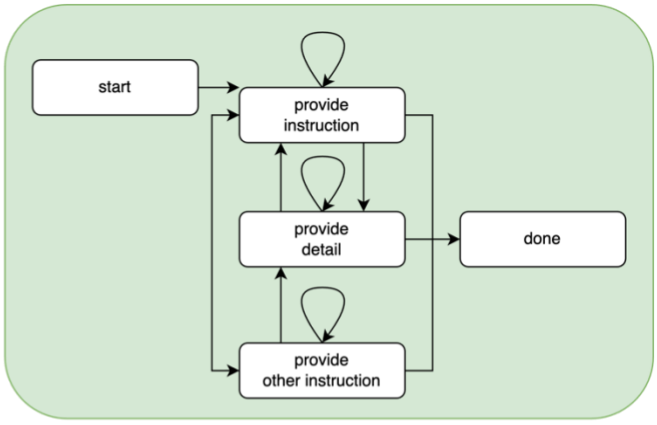
In this work, we proposed a user profile module that stores the information of the conversation progress and domain knowledge (Figure 3). Employing user profile module has multiple benefits. First, the user profile module preserves individual conversation states, enabling simultaneous interactions with multiple users while tracking each user's progress and generating contextually appropriate responses. Second, users can interact with chatbot configured with different domains or dialogue flows, providing flexibility in handling diverse types of conversations. Third, the chatbot can respond according to the user proficiency level to enhance effectiveness of the conversation and user satisfaction.

3.1.1 Conversation Progress Tracking

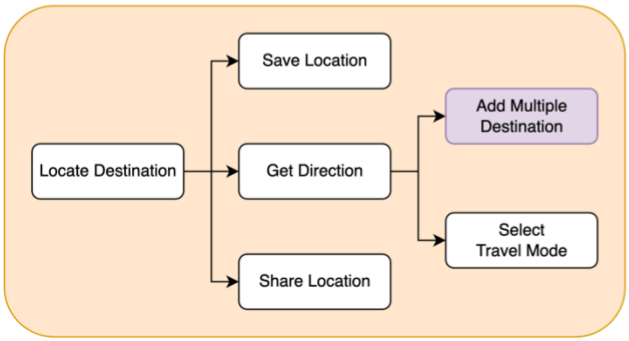
To manage dialogue flow smoothly, a dual-level state machine is implemented to manage conversations. The lower-level state machine is meant to fulfill the purpose of the current conversation defined in the higher-level state machine (Figure 4b). The states are defined as the mission the chatbot ought to complete and the state transition events are defined as speech acts [16]. User’s input would be classified with limited number of possible speech acts and triggers the state transition. The lower-level state machine is considered complete if there are no speech acts defined for state transition.



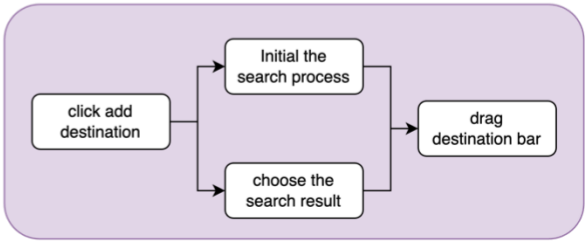
(a) Dialogue Flow Management Unit



(b) Conversational Purpose Contains Speech Acts



(c) Task Flow Management Unit



(d) Method Contains Operators

Figure 4. Structures in User Profile Unit: (a) The dialogue flow management unit is built with multiple conversation purpose units; (b) The conversational units are built with multiple speech act units; (c) The task flow management unit represents the goal, which is the domain knowledge integrated to the chatbot. It is built with multiple methods; (d) All method units can be a goal, yet dependencies exist. Therefore, we separate them as method units. The method units are built with multiple operators, which are the smallest steps that cannot be further divided.

The higher-level state machine manages states represents the entire dialogue flow defined to support the user to perform any tasks on the application (Figure 4a). The states are defined as the purpose of the conversation and the state transition events are defined as user intent. User's input would be classified with limited number of possible intents and triggers the state transition if current lower-level state machine is complete.

The system also maintains error handling when unexpected user input is sent to the dialogue. Additionally, the system allow user to restart the whole conversation.

3.1.2 Domain Knowledge Integration

To accommodate users with limited domain knowledge, we structured tasks using three fundamental components: goals, methods, and operators [13] which are stored as a dual-level state machine. At the basic level, operators represent atomic actions that cannot be further decomposed, such as button clicks or text input. Methods are constructed of operators (Figure 4d) such as saving location of a place or getting the route to a certain place. Goals represent the objectives that users aim to achieve (Figure 4c), which is the series of methods that could lead to that outcome and each method could be the end of the goal.

This hierarchical organization of domain knowledge enables the dialogue system to collaboratively construct and refine goals with users through interactive conversations. The system engages users through targeted questions about potential goals and analyzes their responses to determine goal alignment. Goal exploration follows a dependency-constrained randomization approach, continuing until either the appropriate goal is identified, or all viable options are exhausted.

The system maintains individual proficiency scores for each operator by monitoring user inputs and task executions. This granular proficiency tracking at the operator level enables the dialogue system to adaptively adjust its response granularity, providing either detailed guidance for operators where users show low proficiency or concise instructions for operators where users demonstrate expertise.

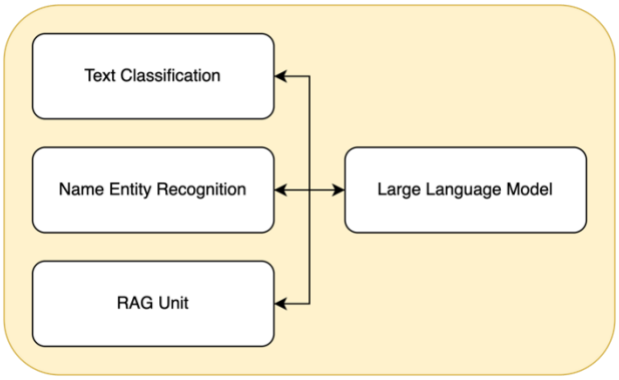


Figure 5. Natural Language Processing Unit

3.2 Natural Language Processing

Pretrained LLMs have showed strong capabilities in performing classification jobs [14], especially in cases with limited training data or when generalization across multiple domains is required. Recent achievement in RAG also gave strength to chatbots to generate responses that are more accurate and specific. Therefore, response generations are not limited to pre-defined replies that are hard coded by humans. RAG also reduces hallucination generated by general diffusion models. Considering portability, privacy, and generalizability, the system integrates local LLM such as LLaMA3 [8] hosted on the Ollama platform.

In the text classification unit, there are indeed two types of text classification that are used. The first type is the task classification that classifies user input into goals that could be performed. The classified goal acts as the necessary information from the user input to support task flow management module to manipulate the goal and methods for further guidance to the user. The second type is the text classification that classifies user input into speech acts or user intents for dialogue flow state transition.

Name entity recognition unit is integrated to extract specific details such as names, location, or entities. These entities could be further used to communicate with the user in the response generation. Additionally, both text classification unit and name entity recognition unit adopted few-shot learning and some prompt engineering to get better result out of low parameter LLM models.

Lastly, RAG is integrated to generate human-like responses without hard coding all the replies. The domain knowledge was to support LLM to generate specific information and avoid hallucination while state statuses both in dialogue flow and task flow help narrow down the information LLM need to consume and generate responses in the specific area defined. By integrating RAG, the system achieves high degree of adaptability and accuracy.

3.3 Dialogue Engine

The dialogue engine works as the central conductor of the system. It manages input and output with the user interface, maintains state transitions, and tracks goal progress. By leveraging a dual-level state machine architecture, the dialogue engine ensures seamless integration of conversational flow and task flow.

To enhance scalability and portability, the dialogue engine communicates with each modules with APIs defined in dialogue flow management module and task flow management module. The modular design kept the domain knowledge out of the core

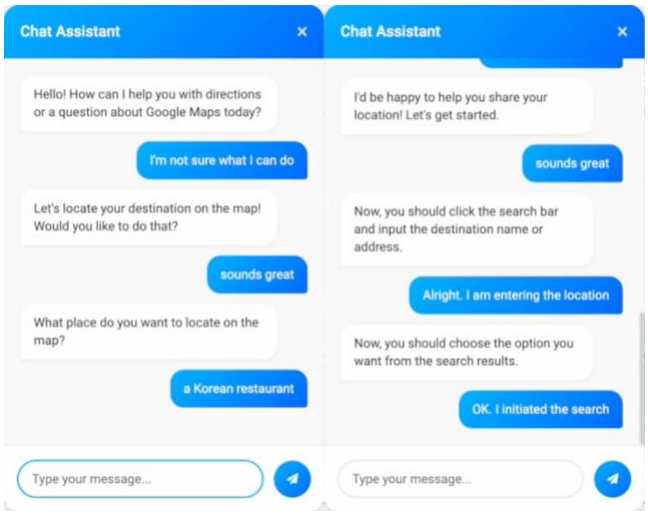


Figure 6. Chat examples Left: chatbot works with the user to set up a goal for the user. Right: chatbot giving instruction or details to support user on task execution.

architecture in the dialogue engine, allowing scalability and adaptability across diverse applications. For example, the chatbot can easily support user a totally different chatbot by replacing the task flow stored in the system.

Additionally, the dialogue engine tracks user proficiency and adapts its response accordingly. The dialogue engine can redirect the state transition to "provide detail" if user proficiency level is detected to be lower than the threshold. By customizing the response, the dialogue engine makes the entire conversation user-centered rather than raising the difficulty to perform the tasks. Users can also benefit from detailed instructions and gradually pick up the knowledge of how to execute the brief instruction recommended by the chatbot. This ensures users receive sufficient support without unnecessary repetition or information overload.

The combination of state-driven architecture, modular domain knowledge integration, and advance natural language processing techniques ensures robust, adaptive, and efficient interaction with the user. This design not only boosts user satisfaction but also supports a wide range of applications requiring customization and context-aware assistance.

4. User Interface Interaction

The chatbot is designed to interact with users through any applications that can invoke API calls. This includes website integrating chat windows and voice interaction with additional voice to text and text to voice unit. The interaction system is built upon two fundamental API methods: initialization and chat.

The initialization method is triggered when users activate the chat function such as clicking on a chatbot icon to start a chat with the chatbot in a small window. A user ID is assigned to the user when the chat is started.

The chat method is an API to support communication from the frontend to the backend. Besides directly typing texts in the chat window and send, the UI we developed and tested on also includes interface-driven communication. Actions performed on the applications are tokenized as operators. They would invoke the chat API after performing the action and send the predefined text containing the information of the operator to match what the

task flow accepts to transit through states. Simultaneously, users can see that there is an extra sentence added to the chat window to notify the user that the system has sent out relevant messages on their behalf. This eliminates the need for users to explicitly confirm their actions with the chatbot. This also avoids user sending something irrelevant to task execution or sent incorrect information that leads to the wrong outcome.

The streamlined approach maintains interaction accuracy while reducing the cognitive load traditionally associated with chatbot confirmation steps. The system design allows future expansion of additional interaction types or interface types.

4.1 Example

Imagine a technology rookie wanted to visit a Korean restaurant and he found a map website that he was unfamiliar with (Figure 1). Upon noticing the chatbot feature, he decided to ask for its assistance. The chatbot helped Jack establish his goal by suggesting possible tasks that could be performed on the website and confirmed his intentions through his response (Figure 6 Left). After establishing the goal, the system had stored the necessary methods in order of their dependencies. Finally, the chatbot guided Jack with instructions and details if was requested (Figure 6 Right) according to the stored methods. The system also sent out message on Jack's behalf when executing on the website. As a result, Jack found his way to the Korean restaurant and enjoyed a great meal.

5. REFERENCES

[1] Oscar D. Andrade, Nathaniel Bean, and David G. Novick. 2009. The macro-structure of use of help. In Proceedings of the 27th ACM international conference on Design of communication (SIGDOC '09). Association for Computing Machinery, New York, NY, USA, 143–150. https://doi.org/10.1145/1621995.1622022

[2] Kimia Kiani, George Cui, Andrea Bunt, Joanna McGrenere, and Parmit K. Chilana. 2019. Beyond "One-Size-Fits-All": Understanding the Diversity in How Software Newcomers Discover and Make Use of Help Resources. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). Association for Computing Machinery, New York, NY, USA, Paper 340, 1–14. https://doi.org/10.1145/3290605.3300570

[3] David G. Novick, Oscar D. Andrade, Nathaniel Bean, and Edith Elizalde. 2008. Help-based tutorials. In Proceedings of the 26th annual ACM international conference on Design of communication (SIGDOC '08). Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/1456536.1456538

[4] David G. Novick and Karen Ward. 2006. Why don't people read the manual? In Proceedings of the 24th annual ACM international conference on Design of communication (SIGDOC '06). Association for Computing Machinery, New York, NY, USA, 11–18. https://doi.org/10.1145/1166324.1166329

[5] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The vocabulary problem in human-system communication. Commun. ACM 30, 11 (Nov. 1987), 964–971. https://doi.org/10.1145/32206.32212

[6] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. 2009. A survey of software learnability: metrics, methodologies and guidelines. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09). Association for Computing Machinery, New York, NY, USA, 649–658. https://doi.org/10.1145/1518701.1518803

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 159, 1877–1901.

[8] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. (February 2023). Retrieved from https://arxiv.org/abs/2302.13971

[9] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23). Association for Computing Machinery, New York, NY, USA, Article 437, 1–21. https://doi.org/10.1145/3544548.3581388

[10] Anjali Khurana, Hariharan Subramonyam, and Parmit K Chilana. 2024. Why and When LLM-Based Assistants Can Go Wrong: Investigating the Effectiveness of Prompt-Based Interactions for Software Help-Seeking. In Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI '24). Association for Computing Machinery, New York, NY, USA, 288–303. https://doi.org/10.1145/3640543.3645200

[11] Satyam Dwivedi, Sanjukta Ghosh, and Shivam Dwivedi. 2023. Breaking the Bias: Gender Fairness in LLMs Using Prompt Engineering and In-Context Learning. Rupkatha Journal on Interdisciplinary Studies in Humanities 15, 4 (2023). https://doi.org/10.21659/rupkatha.v15n4.10

[12] Kotaro Shukuri, Ryoma Ishigaki, Jundai Suzuki, Tsubasa Naganuma, Takuma Fujimoto, Daisuke Kawakubo, Masaki Shuzo, and Eisaku Maeda. 2023. Meta-control of Dialogue Systems Using Large Language Models. (December 2023). Retrieved from https://arxiv.org/abs/2312.13715

[13] S. Card, Allen Newell, and Thomas P. Moran. The Psychology of Human-Computer Interaction. 625. https://doi.org/10.1201/9780203736166

[14] Wang, Z., Pang, Y., & Lin, Y. (2023). Large Language Models Are Zero-Shot Text Classifiers. *arXiv.Org*, *abs/2312.01044*. https://doi.org/10.48550/arxiv.2312.01044

[15] Andrew Freed, Conversational AI: Chatbots that work, Manning, 2021., 3–16

[16] Maite Taboada. 2003. Modeling Task-Oriented Dialogue. *Computers and The Humanities* 37, 4 (November 2003), 431–454. https://doi.org/10.1023/A:1025729107628

[17] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. *Association for Computational Linguistics*. Retrieved from https://aclanthology.org/D18-1547/

[18] Cristen Torrey, Susan Fussell, and Sara Kiesler. 2013. How a robot should give advice. In Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction (HRI '13). IEEE Press, 275–282.

[19] Maite Taboada. 2003. Modeling Task-Oriented Dialogue. *Computers and The Humanities* 37, 4 (November 2003), 431–454. https://doi.org/10.1023/A:1025729107628

[20] Dan Bohus and Alexander I. Rudnicky. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language* 23, 3 (July 2009), 332–361. https://doi.org/10.1016/J.CSL.2008.10.001

[21] Chi-Hsun Li, Su-Fang Yeh, Tang-Jie Chang, Meng-Hsuan Tsai, Ken Chen, and Yung-Ju Chang. 2020. A Conversation Analysis of Non-Progress and Coping Strategies with a Banking Task-Oriented Chatbot. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376209

[22] Asbjørn Følstad, Effie L.-C. Law, and Nena van As. 2024. Conversational Breakdown in a Customer Service Chatbot: Impact of Task Order and Criticality on User Trust and Emotion. ACM Trans. Comput.-Hum. Interact. 31, 5, Article 66 (October 2024), 52 pages. https://doi.org/10.1145/3690383

[23] Amon Rapp, Arianna Boldi, Lorenzo Curti, Alessandro Perrucci, and Rossana Simeoni. 2023. Collaborating with a Text-Based Chatbot: An Exploration of Real-World Collaboration Strategies Enacted during Human-Chatbot Interactions. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3544548.3580995