

HierTOD: A Task-Oriented Dialogue System Driven by Hierarchical Goals

Lingbo Mo
The Ohio State University
mo.169@osu.edu

Shun Jiang
Adobe Inc.
shunj@adobe.com

Akash Maharaj
Adobe Inc.
maharaj@adobe.com

Bernard Hishamunda
Adobe Inc.
hishamun@adobe.com

Yunyao Li
Adobe Inc.
yunyao@adobe.com

ABSTRACT

Task-Oriented Dialogue (TOD) systems assist users in completing tasks through natural language interactions, often relying on a single-layered workflow structure for slot-filling in public tasks, such as hotel bookings. However, in enterprise environments, which involve rich domain-specific knowledge, TOD systems face challenges due to task complexity and the lack of standardized documentation. In this work, we introduce HierTOD, an enterprise TOD system driven by *hierarchical* goals that can support *composite* workflows. By focusing on goal-driven interactions, our system serves a more proactive role, facilitating mixed-initiative dialogue and improving task completion. Equipped with components for natural language understanding, composite goal retriever, dialogue management, and response generation, backed by a well-organized data service with domain knowledge base and retrieval engine, HierTOD delivers efficient task assistance as judged by human evaluators. Furthermore, our system implementation unifies two TOD paradigms: slot-filling for information collection and step-by-step guidance for task execution. Our user study demonstrates the effectiveness and helpfulness of HierTOD in performing both paradigms.

VLDB Workshop Reference Format:

Lingbo Mo, Shun Jiang, Akash Maharaj, Bernard Hishamunda, and Yunyao Li. HierTOD: A Task-Oriented Dialogue System Driven by Hierarchical Goals. VLDB 2025 Workshop: Data Science with Human-in-the-loop.

VLDB Workshop Artifact Availability:

The demo of HierTOD system is available at https://www.youtube.com/shorts/_v8f3x5IKew.

1 INTRODUCTION

Task-oriented dialogue (TOD) systems aim to help users accomplish specific goals by executing tasks through natural language interactions. Significant advancements have been made in developing systems that support public domain tasks [2, 12, 14], such as booking hotels or reserving restaurants. These tasks typically feature straightforward, single-layered workflows with well-defined

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment. ISSN 2150-8097.

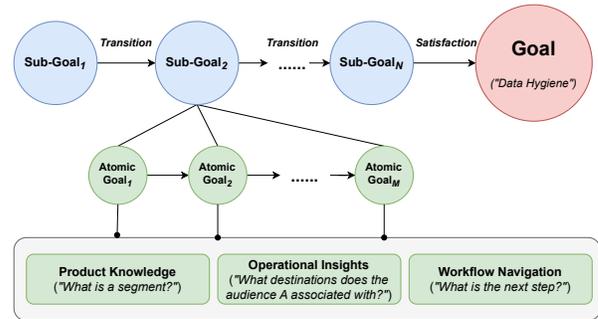


Figure 1: Composite workflows to execute tasks for hierarchical goal completion. A high-level goal consists of multiple sub-goals that transition between each other as the conversation progresses. Each sub-goal can include atomic goals such as product knowledge QA, operational insights QA, or navigation instructions.

steps, intents, and information extraction requirements. However, in enterprise environments rich with domain-specific knowledge, TOD systems face unique challenges due to the complexity and lack of standardized documentation of tasks.

Enterprise tasks often involve multi-layered structures composed of numerous interconnected subtasks, as illustrated in Figure 1. These tasks are rarely formally documented, and their execution heavily relies on the implicit knowledge of human experts. When users interact with a dialogue system in such contexts, their utterances can pertain only to atomic tasks defined at the leaf nodes of a complex task hierarchy. The overarching structure and sequence of these tasks remain internalized within the user's expertise, making it difficult for dialogue systems to fully comprehend and assist effectively.

Consider, for instance, an enterprise marketing platform equipped with audience segmentation functionality. Users within an organization may independently create many client profile segments for various projects. Over time, this practice can result in the proliferation of duplicate segments, increasing platform resource costs and reducing business efficiency. To mitigate this, users perform data hygiene processes to clean up and consolidate these duplicate segments. However, the formal steps for this process are typically undocumented, and practices vary across different organizations. With guidance from subject matter experts, the complex task of

identifying and removing duplicate segments can be distilled into a high-level goal comprising four sequential steps:

Goal: Data Hygiene for Audience Segments

- Step 1: Detect duplicate segments by definition or outcome.
- Step 2: List segment references by relevant business entities.
- Step 3: Remove or unlink non-essential segment references and relink to essential ones when necessary.
- Step 4: Delete non-essential segments.

In practice, different users may prioritize certain business entities over others and follow various paths through this task hierarchy when interacting with the dialogue system. Their intents correspond to atomic actions represented as nodes within a task graph, and the sequences connecting these nodes can vary significantly. A dialogue system with a deep understanding of these high-level business goals and the complex structure of such tasks can significantly enhance user experience. It can improve query comprehension, disambiguate user intents more effectively, proactively suggest relevant goals, and provide personalized responses that align with individual user needs.

To this end, we introduce HierTOD, an enterprise TOD system driven by hierarchical goals to facilitate the generation of more proactive and effective dialogues with users. For example, when a user inquires about detecting duplicate segments (as described in the first step of the previous case), HierTOD can proactively suggest transitioning to the high-level goal of conducting data hygiene for audience segments. Our system comprises several modules, including natural language understanding, composite goal retriever, dialogue management, and response generation, supported by a well-organized data service with a domain knowledge base and retrieval engine.

Furthermore, existing TOD systems typically follow one of two paradigms, which are often developed separately. The first is slot-filling for information collection [6, 7, 16], where users provide details and direct the system to perform specific tasks, such as making reservations. The second paradigm is step-by-step guidance, designed to assist users in executing real-world tasks by providing accurate information and step-by-step instructions. For instance, Amazon’s Alexa Prize Taskbot [1, 4] helps users complete tasks such as cooking a dish or following a DIY tutorial, guiding them through the process with detailed instructions [10]. In this work, HierTOD serves a unified framework that supports both paradigms, delivering comprehensive and efficient task assistance. We implement HierTOD and integrate it into a prototype version of the AI Assistant [9] within the Adobe Experience Platform¹.

Our contributions include: (1) Developing a TOD system to support composite workflows in enterprise environments. (2) Introducing a goal-driven approach to dialogues, making the system more proactive and enabling mixed-initiative interactions for improved task completion. (3) Implementing a unified framework that integrates two TOD paradigms: slot-filling for information collection and step-by-step guidance for task execution. (4) Conducting a user study to verify the effectiveness and helpfulness of our dialogue system.

¹<https://experienceleague.adobe.com/en/docs/experience-platform/ai-assistant/home>

2 RELATED WORK

Task-Oriented Dialogue (TOD) systems are designed to help users accomplish goals through conversational interactions. TOD systems are typically categorized into pipeline-based and end-to-end approaches. Pipeline-based methods decompose the dialogue process into modular components, including natural language understanding, dialogue state tracking, dialogue policy, and natural language generation, with each module handling a distinct subtask [3, 13]. In contrast, end-to-end methods integrate these components into a unified model trained jointly to optimize response generation [5, 16].

The emergence of large language models (LLMs) pretrained on open-domain data has significantly improved performance across various TOD modules [8, 15, 17]. These models can generalize to unseen tasks with minimal supervision through zero-shot or few-shot learning, reducing reliance on extensive annotated datasets. However, while LLMs excel in generic domains, they often struggle in enterprise settings, which involve closed-source knowledge and complex task structures. To address this, HierTOD is proposed as an enterprise TOD system driven by hierarchical goals. It is designed to support composite workflows by integrating enterprise-specific knowledge bases and a composite goal retriever.

Furthermore, traditional TOD systems are largely user-driven, where the user provides structured input to complete tasks such as hotel reservations or flight bookings [2, 12, 14]. In contrast, systems developed for the Alexa Prize TaskBot Challenge[1, 4] focus on interactive task guidance, helping users complete real-world activities like cooking or DIY projects by delivering step-by-step instructions[10]. HierTOD aims to unify these two paradigms, including both slot-filling for information collection and instructional guidance for task execution within one single framework, enabling more flexible and robust dialogue support across a wide range of user goals.

3 SYSTEM DESIGN

3.1 System Overview

HierTOD follows a canonical pipeline approach for TOD systems. The system consists of Natural Language Understanding (NLU), Dialogue Management (DM), and Response Generation (RG). A key feature of our system is the introduction of a Composite Goal Retriever (CGR) which builds a goal repository to define and store workflows for goal completion. Upon receiving user input, the NLU module (Section 3.2) preprocesses the utterance to determine the user’s intent. The CGR module (Section 3.3) then matches the user query against the workflows in the goal repository to identify the appropriate dialogue paradigm, whether slot-filling for information collection or step-by-step guidance for task execution. Once a goal is initiated, the DM module (Section 3.4), which is designed as a hierarchical finite state machine, controls the dialogue flow, handles exceptions, and guides the conversation toward task completion. The RG module (Section 3.5) then generates responses or answers user queries based on intent and dialogue history. It is supported by a well-organized data service, including a domain-specific knowledge base and retrieval engine, which connects to various sources to provide optimal user assistance. We describe the details for each module in the following sections.

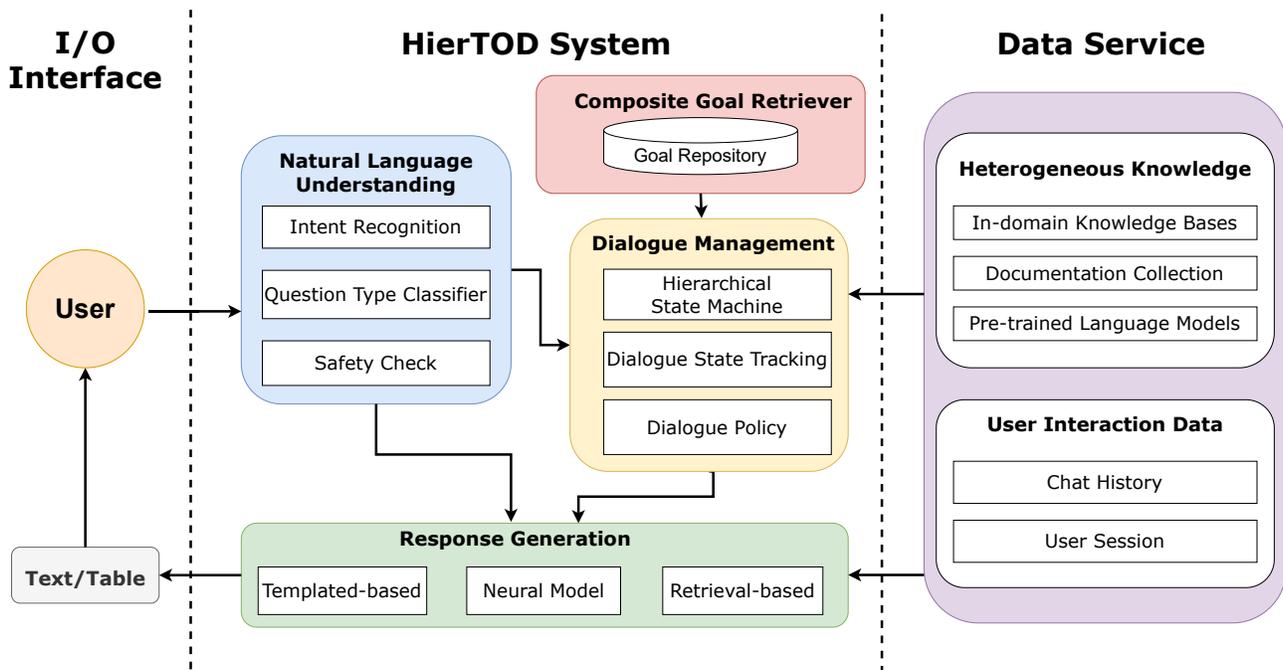


Figure 2: Architecture Overview. HierTOD consists of four major components: Natural Language Understanding, Composite Goal Retriever, Dialogue Management, and Response Generation. These components are supported by a data service that integrates heterogeneous knowledge and has access to user interaction data.

3.2 Natural Language Understanding

HierTOD employs a robust NLU module which combines the strengths of both trained models and rule-based approaches. The key component is Intent Recognition, where we organize multiple intents into three categories to accommodate different user initiatives, as detailed in Table 1. To categorize different types of queries, we train a three-way classification model to classify user queries as either product knowledge, operational insights, or out-of-scope questions. For other intents, we utilize heuristics and keyword lists for recognition.

3.3 Composite Goal Retriever

We define the workflow for hierarchical goal completion in our dialogue system, as illustrated in Figure 1. Specifically, a *high-level goal* consists of multiple *sub-goals* that can transition from one to another based on how the conversation proceeds. Each sub-goal may involve various types of QA interactions and navigation instructions. To this end, we establish and maintain a repository that defines workflows for various goals using a YAML structure. Each workflow includes a high-level goal description, followed by a series of steps to achieve that goal. Each step contains a summary, along with a detailed explanation of the actions required for completion. Based on the user’s query, we employ a CGR module that matches the query to the goals defined in the repository, using both lexical and semantic matching, to determine whether a goal is triggered,

Category	Description
Sentiment	The user can confirm or reject the system’s response in each turn, resulting in two labels: Acknowledgment and Negation , indicating the polarity of the user’s utterance.
Commands	The user can guide the conversation using commands such as: Goal Trigger , Navigation (e.g., moving to previous or next steps), Task Completion , and Stop to terminate the conversation at any point.
Utilities	The Question intent is used to capture various types of queries, including those related to product knowledge, operational insights, or out-of-scope questions.

Table 1: Categories of detailed intents to support diverse user initiatives.

and if so, which goal is activated. If no goal is matched, the system defaults to the standard dialogue mode, focusing on question answering and general conversation.

Additionally, the goal repository accommodates both dialogue paradigms: slot-filling for information collection and step-by-step guidance for task execution. To distinguish between the two paradigms, we introduce the keyword ‘slots’ to specify the information required

for task completion. For example, when the user would like to create a ticket on the enterprise platform for assistance, the necessary slots might include ‘ticket title’, ‘detailed ticket description’, ‘priority’, and ‘phone number’. This repository can be easily expanded or modified by domain knowledge experts to support new goals.

3.4 Dialogue Management

We design a hierarchical finite state machine for the dialogue management component, consisting of two phases: *Goal Pending* and *Goal Execution*. Each phase contains multiple fine-grained dialogue states. In the Goal Pending phase, users interact with HierTOD by issuing a query. If the query matches a high-level goal in the Goal Repository (e.g., “How to perform data hygiene to delete duplicate audience segments?”), the system provides an overview of the task, summarizing the upcoming steps, and then enters into the Goal Execution phase to guide the user step by step.

If the query matches a sub-goal (e.g., “List the duplicate segments for me.”), the system provides an answer to the sub-goal and proactively asks a clarifying question for goal transition, such as, “Would you like to delete the duplicate segments?” If the user agrees to pursue the proposed high-level goal, they move to the Goal Execution phase until task completion. The initial step related to the matched sub-goal is skipped, as it has already been addressed. When the user query does not trigger any goal from the repository, the system utilizes the QA module to provide an appropriate response. Additionally, during the Goal Execution phase, the QA module remains available to support the user with relevant questions.

Dialogue State Tracking. For the step-by-step guidance dialogue paradigm, we use the hierarchical state machine mentioned above to keep track of the states throughout the conversation. In order to further support the slot-filling paradigm, we employ zero-shot learning using GPT-3.5 [11] to perform dialogue state tracking (DST). In the designed prompt, general instructions are provided to capture values for the required slots based on both the dialogue history and the current user utterance (see Appendix A.1 for the exact prompt). The updated belief state is then utilized in the subsequent response generation component described in Section 3.5.

Dialogue Policy. The dialogue policy takes inputs from the NLU and CGR modules, synchronizes with the hierarchical state machine to query and update the task step state for step-by-step guidance dialogues. It also interacts with the DST to manage and update the task-specific belief state for slot-filling dialogues. The policy then drives the RG module to generate an appropriate response to the user’s utterance, which will be described in the next section.

3.5 Response Generation

Our response generation module blends both infilling-based methods and neural models. For the step-by-step guidance dialogue paradigm, we use handcrafted conditional rules to organize curated templates and define their composition strategy according to the high-level states in our hierarchical state machine. For the slot-filling dialogue paradigm, we utilize the belief state from the DST module and make an LLM call to GPT-3.5. This can generate either a question to request missing slots or a summary to conclude the task when all required slots have been filled. The designed prompt

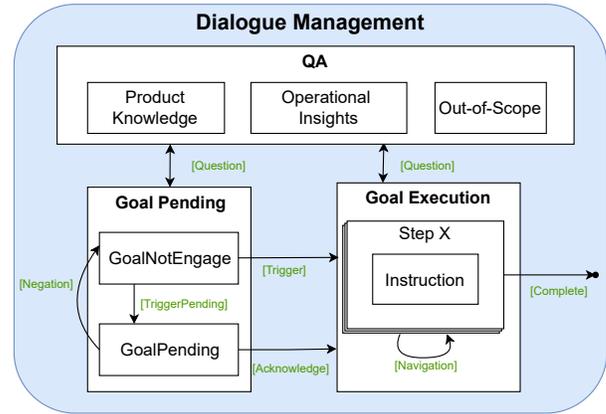


Figure 3: Dialogue Management Diagram consisting of two phases: Goal Pending and Goal Execution. White boxes represent dialogue states, while green text in brackets indicates user intent.

includes the task description, belief state with filled and missing slots, dialogue history, and current user utterance, instructing the model to generate an appropriate response (see Appendix A.2 for the exact prompt).

In parallel, we develop a QA module to provide answers when users have questions during goal execution. As mentioned earlier, we first implement a routing model—a three-way classifier—that categorizes user questions into product knowledge, operational insights, or out-of-scope inquiries. For out-of-scope questions, we provide a predefined template response. We follow the AI Assistant [9] within the Adobe Experience Platform for the QA module implementation. The handling of the first two types of questions is detailed in the following parts.

Product Knowledge QA. Product knowledge refers to concepts and topics grounded in the product documentation. Product knowledge questions can be further specified into the following sub-groups, including pointed learning, open discovery, and troubleshooting. The Product Knowledge QA component identifies the relevant documentation to answer a given question, retrieves the appropriate content, generates a response based on the retrieved information, determines proper source citations, and verifies that the responses are well-grounded.

Operational Insight QA. Operational insights refer to the information about metadata objects, such as attributes, audiences, dataflows, datasets, destinations, journeys, schemas, and sources including counts, lookups, and lineage impact. For example: “How many datasets do I have?” or “How many schema attributes have never been used?” The Operational Insights QA component translates user questions into SQL queries against the underlying operational data specific to customers, executes the queries to generate accurate responses, and provides comprehensive explanations for both the query and the resulting answer. These explanations include: (1) query interpretation, offering a concise summary of the

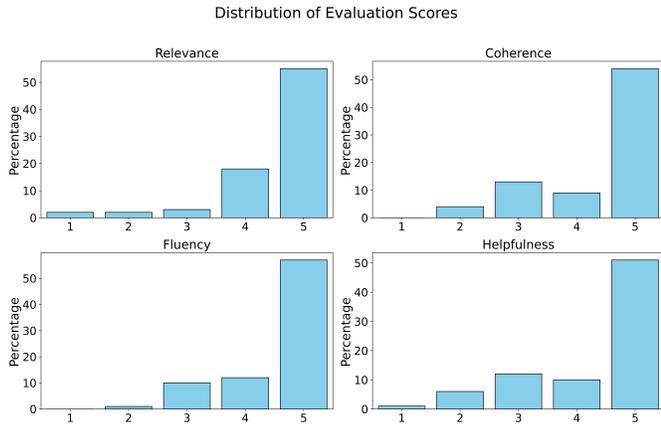


Figure 4: Distribution of evaluation scores across four aspects in the user study.

	Relevance	Coherence	Fluency	Helpfulness
Average Score	4.37	4.31	4.46	4.20

Table 2: User study results based on a 5-point Likert scale, evaluating four aspects: Relevance, Coherence, Fluency, and Helpfulness.

actions taken to address the user’s question; (2) a step-by-step breakdown of how the query is processed; and (3) inline SQL comments, allowing users with strong technical backgrounds to verify the SQL logic. These elements not only enhance transparency but also empower users to gain deeper insights into the underlying data processes.

3.6 NL2Goal Translator

To further simplify the goal workflow creation process, we develop an automatic NL2Goal translator powered by in-context learning using GPT-3.5. This module takes a manually crafted goal description in natural language and translates it into a structured goal definition, which is then stored in the composite goal repository in YAML format. By automating the creation of composite goals, this approach enhances the flexibility to expand and modify the goal repository, making it more feasible to adopt our goal-driven dialogue generation system in knowledge-rich domains at enterprise scale. The exact prompt for this module can be found in Appendix A.3.

4 USER STUDY

To evaluate the performance of our dialogue generation system, we conduct a user study involving five annotators, each with a background in machine learning and experience in developing conversational AI systems. Every annotator is tasked with reviewing the same set of 20 dialogues between a user and an AI assistant.

These dialogues cover a variety of tasks, including product platform operations, troubleshooting issues, and general activities such as booking a restaurant or hotel.

In the user study, annotators are asked to rate each dialogue across four aspects: relevance, coherence, fluency, and helpfulness, defined as follows.

- **Relevance:** Assesses how accurately the HierTOD system’s responses align with and address the user’s queries.
- **Coherence:** Evaluates the logical flow and consistency of the HierTOD system’s responses throughout the conversation. Specifically, it measures how well each reply is contextually appropriate, connected to previous exchanges, and aligned with the user’s goal, without introducing contradictions or abrupt shifts in topic.
- **Fluency:** Evaluates the grammatical correctness and naturalness of the HierTOD system’s language.
- **Helpfulness:** Measures the overall helpfulness of the HierTOD system that enables the user to complete their task.

Among these aspects, Relevance and Fluency focus on individual interaction turns, while Coherence and Helpfulness are assessed based on the entire dialogue. Detailed descriptions of these four aspects provided to the annotators can be found in Appendix B. Each aspect is rated on a 5-point Likert scale, where 1 indicates the lowest score (e.g., not relevant at all) and 5 indicates the highest score (e.g., completely relevant). The annotation interface allows the annotators to view a screenshot of the conversation and rate each aspect, with optional comment sections to provide further insights or clarify their ratings.

The main results are shown on Table 2, which includes the average evaluation scores from five annotators. The results show that HierTOD performs well across all four aspects, with average scores above 4. The detailed distribution of evaluation scores, shown in Figure 4, provides further insight into the system’s performance. Most of the scores are concentrated at 4 or 5. However, for Coherence and Helpfulness, there is a noticeable presence of scores at 2 or 3. Since both aspects assess the quality of the entire dialogue, this suggests that maintaining consistent performance throughout a conversation remains a challenge, particularly as the conversation length increases.

5 CASE STUDY

We conduct a case study in this section to better understand the strengths and limitations of the system. As illustrated in the dialogue example in Figure 5, the user initially asks to list duplicated audiences with the same segment definition. Our HierTOD system successfully executes this request and returns the duplicated audiences. Moreover, it identifies a related sub-goal in the goal repository and transitions to a broader goal to delete the duplicated audiences, actively asking the user whether they would like to proceed. This reflects the system’s ability to think one step ahead, making it more proactive and intelligent.

However, some users in the study rated the system low on helpfulness in this case, providing feedback that for certain steps, such as relinking a duplicated audience from the destinations, it would be more useful if the system directly executed the action rather than offering step-by-step instructions. While our system currently supports some API integrations, such as listing duplicated audiences,

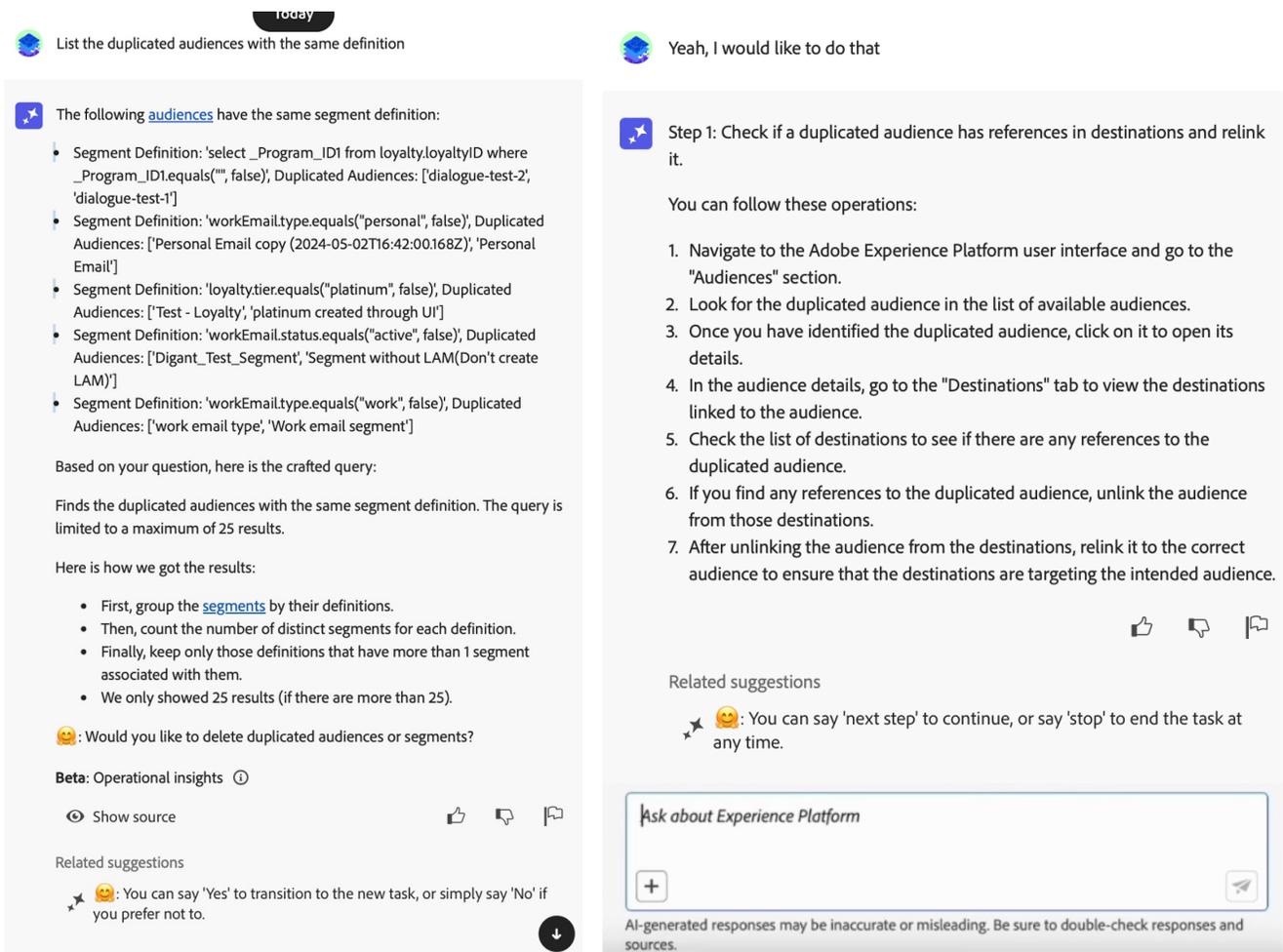


Figure 5: Case study. The first two turns of an example dialogue in which the user asks to list duplicated audiences.

direct execution of internal actions that involve sensitive knowledge or customer data still requires significant effort in fully integrating enterprise-specific toolkits and internal APIs. Nonetheless, this points toward a promising future direction: evolving HierTOD into a more agentic system that can access internal APIs, perform web search, retrieve documentation, and ground its actions in outputs that better fulfill user needs. The interface of our system is provided in Appendix C, along with more dialogue examples that cover both dialogue paradigms, including step-by-step guidance for task execution and slot-filling for information collection.

6 CONCLUSION

In this work, we introduce HierTOD, a modular task-oriented dialogue system designed to assist users in completing tasks within enterprise environments. Our system features a comprehensive set of modules, and adopts a goal-driven approach to dialogues, making it more proactive and enabling mixed-initiative interactions. Furthermore, we implement a unified framework that integrates two representative TOD paradigms including slot-filling for information

collection and step-by-step guidance for task execution. Results from our human study confirm the effectiveness and helpfulness of our dialogue system.

ETHICS STATEMENT

Privacy and safety are top priorities in our dialogue system. We ensure that no personal or identifying information is incorporated into conversations. To further protect users, the system includes a series of safety checks, proactively rejecting inappropriate task requests and preventing the display of potentially harmful tasks that could pose risks to users or their belongings.

ACKNOWLEDGMENTS

We would like to express our gratitude to the team at Adobe for their valuable comments and feedback: Anirudh Sureshan, Pawan Sevak, Jake Bender, Vaishnavi Muppala, Shaista Hussain, Shreya Anantha Raman, Rachel Hanessian, Danny Miller, and Pritom Baruah.

REFERENCES

- [1] Eugene Agichtein, Michael Johnston, Anna Gottardi, Lavina Vaz, Cris Flagg, Yao Lu, Shaohua Liu, Sattvik Sahai, Giuseppe Castellucci, Jason Ingyu Choi, et al. 2023. Advancing conversational task assistance: the second Alexa Prize TaskBot challenge. (2023).
- [2] Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, et al. 2020. Task-Oriented Dialogue as Dataflow Synthesis. *Transactions of the Association for Computational Linguistics* 8 (2020), 556–571.
- [3] Wenhui Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically Conditioned Dialog Response Generation via Hierarchical Disentangled Self-Attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3696–3709.
- [4] Anna Gottardi, Osman Ipek, Giuseppe Castellucci, Shui Hu, Lavina Vaz, Yao Lu, Anju Khatri, Anjali Chadha, Desheng Zhang, Sattvik Sahai, et al. 2022. Alexa, let’s work together: Introducing the first alexa prize taskbot challenge on conversational task assistance. *arXiv preprint arXiv:2209.06321* (2022).
- [5] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems* 33 (2020), 20179–20191.
- [6] Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022. In-Context Learning for Few-Shot Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2627–2643.
- [7] Vojtěch Hudeček and Ondřej Dušek. 2023. Are Large Language Models All You Need for Task-Oriented Dialogue?. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 216–228.
- [8] Zekun Li, Wenhui Chen, Shiyang Li, Hong Wang, Jing Qian, and Xifeng Yan. 2022. Controllable Dialogue Simulation with In-context Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 4330–4347.
- [9] Akash V Maharaj, Kun Qian, Uttaran Bhattacharya, Sally Fang, Horia Galatanu, Manas Garg, Rachel Hanessian, Nishant Kapoor, Ken Russell, Shivakumar Vaithyanathan, et al. 2024. Evaluation and Continual Improvement for an Enterprise AI Assistant. *Data Science with Human-in-the-Loop* (2024), 17.
- [10] Lingbo Mo, Shijie Chen, Ziru Chen, Xiang Deng, Ashley Lewis, Sunit Singh, Samuel Stevens, Chang-You Tai, Zhen Wang, Xiang Yue, et al. 2023. Roll Up Your Sleeves: Working with a Collaborative and Engaging Task-Oriented Dialogue System. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 197–201.
- [11] OpenAI. 2022. ChatGPT. <https://chat.openai.com>.
- [12] Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building Task Bots at Scale with Transfer Learning and Machine Teaching. *Transactions of the Association for Computational Linguistics* 9 (2021), 807–824.
- [13] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2182–2192.
- [14] Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 4661–4676.
- [15] Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and He-Yan Huang. 2024. Rethinking Task-Oriented Dialogue Systems: From Complex Modularity to Zero-Shot Autonomous Agent. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2748–2763.
- [16] Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 14230–14238.
- [17] Jeffrey Zhao, Yuan Cao, Raghav Gupta, Harrison Lee, Abhinav Rastogi, Mingqiu Wang, Hagen Soltau, Izhak Shafran, and Yonghui Wu. 2023. AnyTOD: A Programmable Task-Oriented Dialog System. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 16189–16204.

A PROMPT DESIGN

A.1 Prompt for Dialogue State Tracking

```
Capture entity values from last utterance of the conversation according to examples.
Capture pair "entity:value" separated by colon and no spaces in between.
Format the output in JSON.
If not specified, leave the value empty. Values that should be captured are:
{slots}

### Here is the conversation between user and ai-assistant:
{chat_history}
<<user>>: {current_utterance}

Capture all the entity values based on the conversation above and format the output in JSON:
```

A.2 Prompt for Response Generation

```
You are a task-oriented dialogue system designed to assist users in completing specific tasks such as
booking a hotel or booking a flight. Your goal is to gather all necessary information (slots)
required to complete the task through a series of user interactions. If all required slots are
collected, you should confirm that the task has been completed.

### Task:
{task}

### Filled Slots:
{filled_slots}

### Missing Slots:
{missing_slots}

### Here is the conversation between user and ai-assistant:
{chat_history}
<<user>>: {current_utterance}

### Requirements:
1. If there are remaining slots that need to be filled, generate a polite and contextually appropriate
utterance to request the next missing piece of information from the user. Ask one missing slot at
a time.
2. If all required slots have been filled, briefly summarize all the collected slot information
without asking the user any questions.
3. If the user asks a question, exactly start the placeholder <ANSWER> as the response, followed by a
polite and contextually appropriate utterance to request the next missing piece of information
from the user.

Generate an contextually appropriate user-facing utterance based on the current task, slot information
and the conversation. The generated utterance should be friendly, polite, and positive.

<<ai-assistant>>:
```

A.3 Prompt for NL2Goal Translator

Given a paragraph that describes a specific goal and its associated workflow, your task is to generate a YAML snippet that structures the information into a list of steps. Each step should include a "name" field summarizing the step and a "description" field for explaining additional details. Ensure that the step numbers in the YAML snippet are consistent with the numbers in the workflow.

Example:

I have a goal to resolve an issue where the data pipeline is failing at the transformation stage. The workflow to address this involves the following steps: first, investigate the transformation logic to ensure all mappings and transformations are correct; second, verify that the data sources are providing complete and accurate data; and third, check the pipeline logs for any errors that might indicate issues during the transformation process.

The corresponding YAML should look like this:

```
workflow:
- goal: "Resolve the issue where the data pipeline is failing at the transformation stage."
  steps:
    - name: "Investigate the transformation logic."
      description: "Ensure that all mappings and transformations are correct."

    - name: "Data verification."
      description: "Verify that the data sources are providing complete and accurate data."

    - name: "Check for errors."
      description: "Look for any errors in the pipeline logs that indicate issues during
        transformation."
```

Task:

{new_goal}

Generate the corresponding YAML snippet:

B USER STUDY METRICS

In our user study, annotators are asked to rate each dialogue on a 5-point Likert scale across four aspects: relevance, coherence, fluency, and helpfulness. The detailed scoring criteria for all four aspects, including Relevance, Coherence, Fluency, and Helpfulness, are provided in the Table 3–6.

Score	Relevance
1	The AI assistant’s responses are not relevant at all to the user’s queries.
2	The AI assistant’s responses are mostly irrelevant to the user’s queries.
3	The AI assistant’s responses are moderately relevant to the user’s queries.
4	The AI assistant’s responses are mostly relevant to the user’s queries.
5	The AI assistant’s responses are all relevant to the user’s queries.

Table 3: Detailed scoring criteria for Relevance in the user study.

Score	Coherence
1	The AI assistant’s responses throughout the conversation are not coherent at all.
2	The AI assistant’s responses throughout the conversation are mostly not coherent.
3	The AI assistant’s responses throughout the conversation are moderately coherent.
4	The AI assistant’s responses throughout the conversation are mostly coherent.
5	The AI assistant’s responses throughout the conversation are completely coherent.

Table 4: Detailed scoring criteria for Coherence in the user study.

Score	Fluency
1	The AI assistant’s responses are not fluent at all.
2	The AI assistant’s responses are mostly not fluent.
3	The AI assistant’s responses are moderately fluent
4	The AI assistant’s responses are mostly fluent.
5	The AI assistant’s responses are all fluent.

Table 5: Detailed scoring criteria for Fluency in the user study.

Score	Helpfulness
1	The AI assistant’s responses throughout the conversation are not helpful at all.
2	The AI assistant’s responses throughout the conversation are mostly not helpful.
3	The AI assistant’s responses throughout the conversation are moderately helpful.
4	The AI assistant’s responses throughout the conversation are mostly helpful.
5	The AI assistant’s responses throughout the conversation are completely helpful.

Table 6: Detailed scoring criteria for Helpfulness in the user study.

C INTERFACE

This section presents the user interfaces of the HierTOD dialogue system, prototyped within the Adobe Experience Platform. Figures 6 and 7 show interface screenshots illustrating partial examples of the system’s two dialogue paradigms: step-by-step guidance for task execution and slot-filling for information collection. Full dialogue examples are available in the demo².

²Demo: https://www.youtube.com/shorts/_v8f3x5IKEw

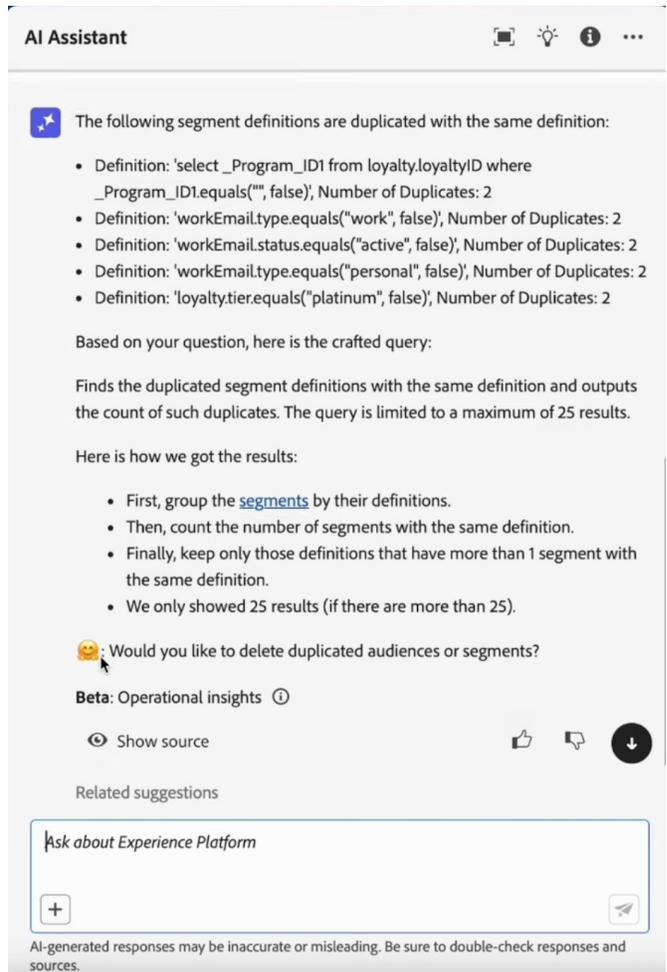
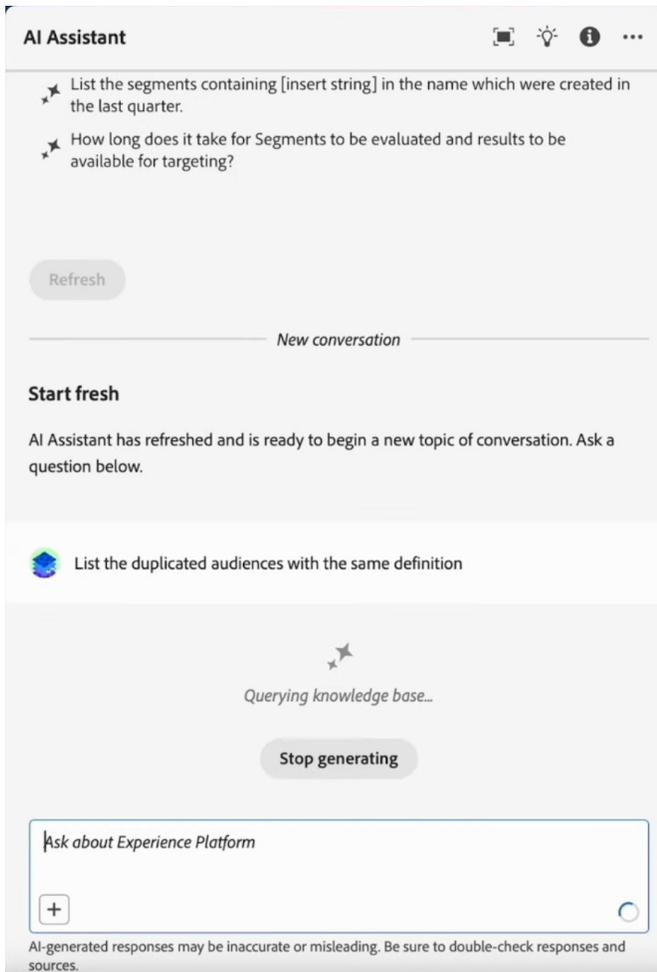


Figure 6: Dialogue A. It follows the step-by-step guidance paradigm for task execution. The example shown focuses on performing data hygiene for audience segments.

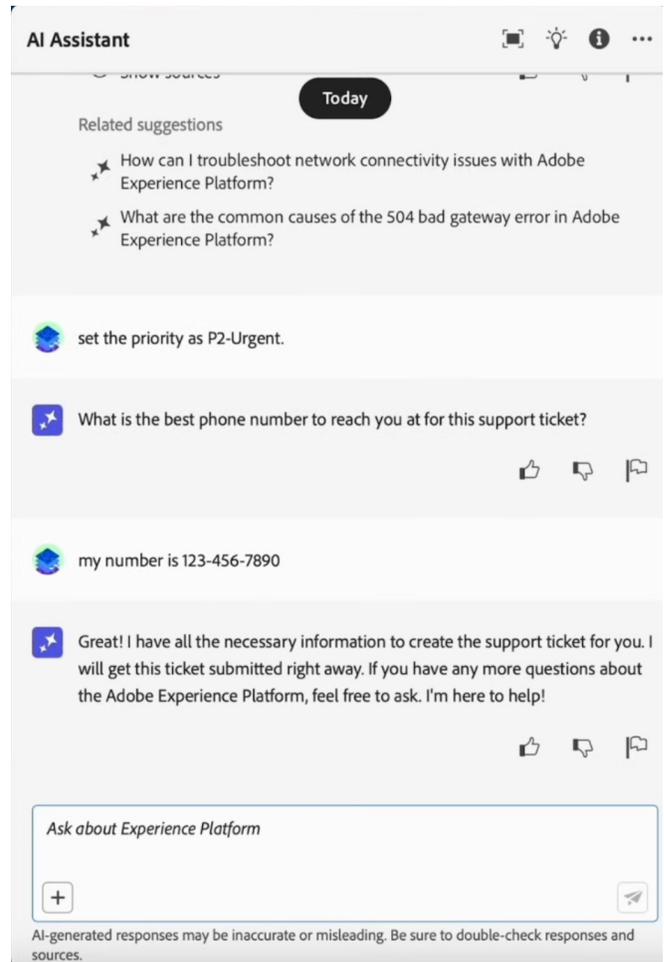
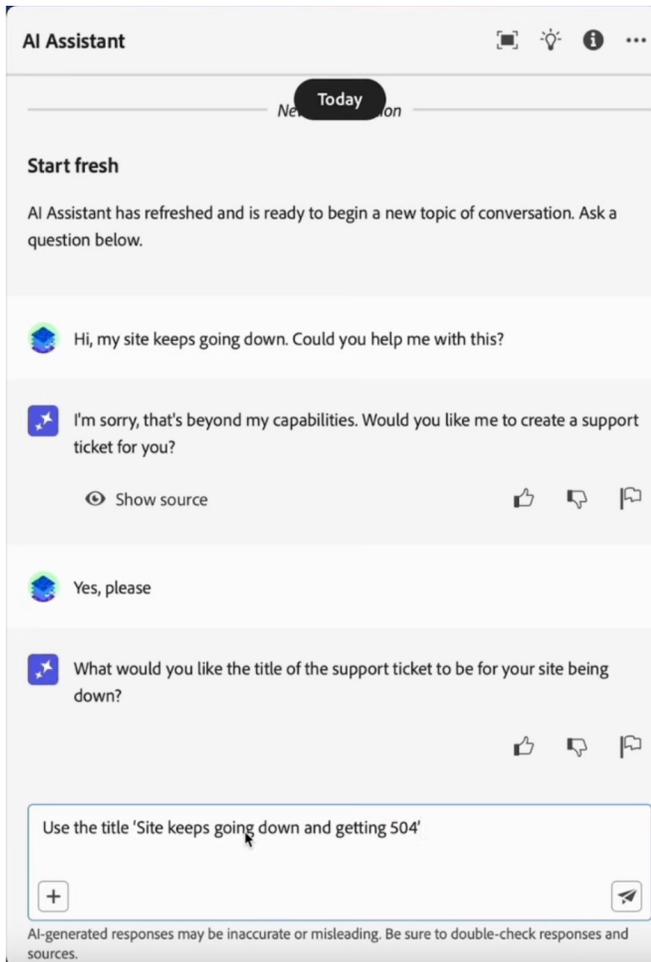


Figure 7: Dialogue B. It follows the slot-filling for information collection paradigm. The example shows the process of helping the user create a support ticket.