Improving Multi-turn Task Completion in Task-Oriented Dialog Systems via Prompt Chaining and Fine-Grained Feedback

Anonymous ACL submission

Abstract

Task-oriented dialog (TOD) systems facilitate users in accomplishing complex, multi-turn tasks through natural language. While traditional approaches rely on extensive fine-tuning and annotated data for each domain, instruction-tuned large language models (LLMs) offer a more flexible alternative. However, LLMs struggle to reliably handle multi-turn task completion, particularly with accurately generating API calls and adapting to new domains without explicit demonstrations. To address these challenges, we propose RealTOD, a novel framework that enhances TOD systems through prompt chaining and fine-grained feedback mechanisms. Prompt chaining enables zero-shot domain adaptation via a two-stage prompting strategy, eliminating the need for human-curated demonstrations. Meanwhile, the fine-grained feedback mechanism improves task completion by verifying API calls against domain schemas and providing precise corrective feedback when errors are detected. We conduct extensive experiments on the SGD and BiTOD benchmarks using four LLMs. RealTOD improves API accuracy, surpassing AutoTOD by 37.74% on SGD and SimpleTOD by 11.26% on BiTOD. Human evaluations further confirm that LLMs integrated with RealTOD achieve superior task completion, fluency, and informativeness compared to existing methods. [1]

1 Introduction

Task-oriented dialog (TOD) systems enable users to accomplish multi-turn tasks, such as booking flights, making restaurant reservations, and managing appointments, through multi-turn, natural language interactions (He et al., 2022). These systems must understand user intent, retrieve relevant information from external systems via API calls, and generate coherent responses to guide users toward task completion. Traditional TOD systems rely on extensive domain-specific fine-tuning and manually annotated datasets, which limit their scalability to new domains and increase deployment costs (Xu et al., 2024; Mi et al., 2022). As a result, developing TOD systems that generalize across diverse domains without extensive supervision remains an open challenge.

Recent advances in instruction-tuned large language models (LLMs) (Chung et al., 2024; Shu et al., 2024) have significantly improved performance across a wide range of natural language processing (NLP) tasks, including text classification (Sun et al., 2023; Wang et al., 2023; Zhang et al., 2024b), summarization (Pu et al., 2023; Zhang et al., 2024a; Van Veen et al., 2023), and response generation (Radford et al., 2019; Brown et al., 2020). These models, trained on diverse instruction-following datasets, can generate fluent and contextually relevant responses in dialog settings (Thoppilan et al., 2022; Dubey et al., 2024; Achiam et al., 2023), making them promising candidates for TOD systems. However, integrating LLMs into TOD remains challenging.

Unlike single-step NLP tasks, TOD systems require multi-turn interactions to collect task-specific information, retrieve external data, and execute user requests while adhering to domain constraints (Chung et al., 2023). While LLMs demonstrate strong generalization in open-ended response generation (Naveed et al., 2023), they struggle with task completion – measured through API call accuracy (Shinn et al., 2023; Jain et al., 2024; Song et al., 2025). Common issues include hallucinated search results, incorrect API method names, invalid slot-value pairs, and missing required parameters, all of which can lead to execution failures and incomplete task fulfillment. Addressing these limitations is critical to unlock the full potential of LLM-based TOD systems.

To overcome these challenges, we propose RealTOD, a novel framework that enhances LLM-

[1] Source code will be released upon acceptance.

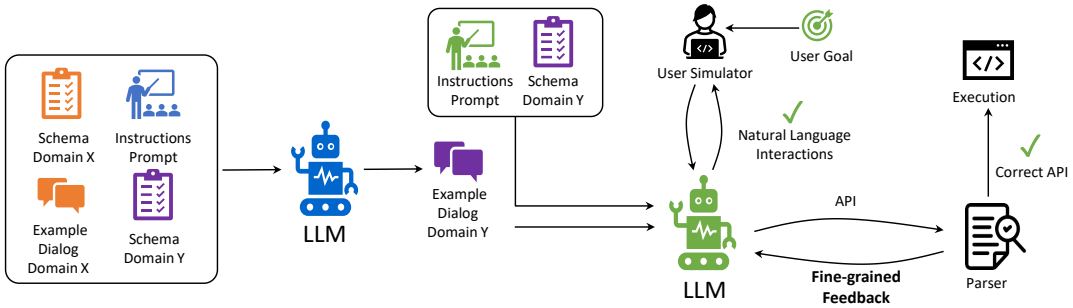

Figure 1: Overview of RealTOD: With only a single example dialog from one domain, RealTOD scales to infinitely many domains through prompt chaining, removing the need for human-curated dialogs for each domain. A fine-grained feedback loop from the API parser further improves API call accuracy.

based TOD systems through prompt chaining and fine-grained feedback. Prompt chaining enables zero-shot domain adaptation by using a two-stage prompting process: first, an example dialog from a source domain is transformed into a target domain dialog by aligning it with the domain schema while maintaining task consistency. This generated example serves as a demonstration, allowing LLMs to generalize to new domains without human-curated examples. Meanwhile, fine-grained feedback improves API execution reliability by systematically verifying API calls against domain schemas, detecting errors (e.g., incorrect method names, invalid slot assignments, missing parameters), and providing precise corrective signals. This iterative feedback mechanism enables real-time error correction, ultimately enhancing task completion rates.

We evaluate RealTOD on two benchmark datasets – SGD (Rastogi et al., 2020) and BiTOD (Lin et al., 2021b) – using four LLMs: two proprietary models (GPT-4o and Claude) and two open-source models (DeepSeek and LLaMA). We measure task completion using full API Call Accuracy, which assesses how often the generated API calls exactly match the ground truth. Even minor errors – such as incorrect method names, missing parameters, or invalid slot-value pairs – can lead to execution failures, resulting in failed tasks. The quality of natural language responses is evaluated using BERTScore (Zhang et al., 2019), which measures semantic similarity between generated and reference responses using contextual embeddings, offering a more reliable assessment than traditional n-gram-based metrics.

Our results show that RealTOD significantly improves API accuracy across all models and datasets. On SGD, it surpasses AutoTOD (Hosseini-Asl et al., 2020) by 37.74% in full API accuracy, while on BiTOD, it outperforms supervised fine-tuned SimpleTOD (Xu et al., 2024) by 11.26%. Human evaluations further confirm that LLMs integrated with RealTOD generate more fluent, informative, and effective task completions than baseline models. Our ablation study demonstrates that both prompt chaining and fine-grained feedback contribute to improved multi-turn dialog quality and reliable task completion.

2 Related Works

Fine-Tuned Task-Oriented Dialog Systems. TOD systems are typically classified into pipeline-based and end-to-end approaches. Pipeline-based methods (Williams and Young, 2007; Lee, 2013; Lee et al., 2009; Peng et al., 2020; Chen et al., 2019) decompose the system into modular components—natural language understanding, dialog state tracking, policy learning, and natural language generation—allowing independent optimization of each module. In contrast, end-to-end approaches (Hosseini-Asl et al., 2020; Madotto et al., 2018; Su et al., 2022; Mosharrof et al., 2023; Siddique et al., 2022; Lei et al., 2018; Lin et al., 2020; Imrattanatrai and Fukuda, 2023) generate responses directly, bypassing these modules. A major drawback of these fine-tuned methods is their reliance on high-quality labeled data, which can be a significant limitation.
LLM-Powered Systems. The rise of LLMs has led to the development of various intelligent systems, which can be broadly categorized into three classes. The first class includes Web Agents, which facilitate online interactions for information retrieval and task execution (Yao et al., 2023; Kim et al., 2024; Ma et al., 2023; Fereidouni et al., 2024; Yao et al., 2022; Sridhar et al., 2023; Furuta et al., 2024). The second class consists of Mobile Agents, which focus on optimizing LLM-based decision-making for performing diverse tasks on mobile applications (Bai et al., 2024; Lee et al., 2023; Wen et al., 2024,

2023; Wang et al., 2024b,a). The third and most relevant class to our work is LLM-powered TOD Systems (Chung et al., 2023; Mi et al., 2022; Gao et al., 2023; Hudeček and Dusek, 2023; Labruna et al., 2023). Specifically, AutoTOD (Xu et al., 2024) shares similarities with our approach; however, AutoTOD does not account for the possibility of LLMs making errors in generating API calls and lacks proper evaluation of API accuracy.

User Simulators. One of the earliest data-driven user simulators is (Eckert et al., 1997), where user actions are generated probabilistically based on system actions. In addition to this, there have been many advancements in data-driven user simulation. For instance, recent advancements leverage transformer-based architectures for domain-independent simulation (Lin et al., 2021a, 2022) and GPT-based models integrating goal state tracking (Liu et al., 2022). Reinforcement learning has also been applied to fine-tune generative simulators (Tseng et al., 2021; Cheng et al., 2022). More recently, in-context learning (ICL) with LLMs has enabled user simulation without fine-tuning, (Terragni et al., 2023; Davidson et al., 2023). Similar to (Lin et al., 2021a, 2022; Liu et al., 2022), our user simulator employs transformer-based architectures.

3 Proposed Framework: RealTOD

We introduce RealTOD, an interactive, real-world TOD framework that eliminates the need for fine-tuning while seamlessly scaling to new domains. RealTOD leverages instruction-tuned LLMs (e.g., Llama) and requires only a single dialog example from any domain to generalize across infinitely many domains using their schemas. At the core of RealTOD is a two-stage prompt chaining and a fine-grained feedback mechanism, enabling it to autonomously request task-related information from the user, generate natural responses, execute API calls, and complete complex, multi-turn tasks – all without domain-specific customization of the LLM. The first stage of prompt chaining generates an example dialog in a new domain by transforming a given source domain dialog into a corresponding target domain example while maintaining task-specific consistency with the target domain schema. The second stage then uses this generated example as an in-context demonstration, enabling the model to adapt to the target domain without requiring additional human-curated dialogs. Additionally, RealTOD integrates a fine-grained feedback mechanism via an API parser that verifies the correctness of API calls. If any errors are detected, the parser provides fine-grained feedback to the LLM for correction, resulting in reliable execution of actions and improved task completion rates.

3.1 Problem Formulation

We formulate multi-turn task completion as a conditional sequence generation problem, where the LLM produces natural language responses or API calls to help users achieve their goals across relevant domains. Each API call includes method name, a dictionary of parameter names and their corresponding values.

Formally, a domain $d_x \in D$ is characterized by a domain schema, which consists of a set of user intents $\mathcal{I}_{d_x}$. An intent represents a specific goal the user aims to achieve in the domain. For example, in the "Flights" domain, an intent might be "Book a Flight". Each intent $i \in \mathcal{I}_{d_x}$ is associated with a set of slots $\mathcal{S}_i$, where each slot s captures relevant constraints to fulfilling the intent. For example, the intent of "Book a Flight" may involve slots such as "departure city" and "destination city". We define a slot s as a tuple:

$$s = (\texttt{name}(s), \texttt{is\_required}(s), \texttt{values}(s))$$

where $\texttt{name}(.)$ specifies the slot's name (e.g., "departure city"), $\texttt{is\_required}(.)$ is a boolean flag indicating whether the slot is mandatory, and $\texttt{values}(.)$ specifies a predefined set of possible values for categorical slots (e.g., "business class", "economy class" in the "Flights" domain). If the slot accepts free-form inputs, this field remains empty. For brevity, we will refer to the name of a slot $\texttt{name}(s)$ as s_m. Formally, the schema for a domain d_x is represented as:

$$\Sigma_{d_x} = (d_x, \mathcal{I}_{d_x}, \{\mathcal{S}_i \mid i \in \mathcal{I}_{d_x}\}).$$

In addition to generating natural language responses, the model may need to retrieve information from external systems or execute actions via API calls to accurately fulfill a user's goal. Each API call corresponds to a specific intent in a domain and a set of specified constraints, represented as slot-value pairs. Formally, an API call a_n is defined as: $a_n = \texttt{API}(\texttt{method} = i, \texttt{parameters} = \{(s_m, v), (\cdots) \mid s_m \in \mathcal{S}_i\})$, where i is the intent, s_m is the slot name, and v is its assigned value. For instance, in the "Flights" domain, an API call for booking a flight may look like:

`API(method` = "Book_a_Flight", `parameters` = {("departure_city", "New York"), ("destination", "London"), $(\cdots)$}).

A dialog session in a domain d_x consists of a sequence of user utterances and system responses across multiple turns. We define a session $\mathcal{T}_{d_x}$ of up to T turns as:

$$\mathcal{T}_{d_x} = \big((u_1, r_1), (u_2, r_2), \ldots, (u_T, r_T)\big)$$

where u_t is the user's utterance at turn t, r_t is the system's response at turn t, which can either be a natural language reply or an API call. The dialog history up to turn t, denoted as H_t, consists of all previous exchanges up to and including the current user utterance: $H_t = \{(u_1, r_1), (u_2, r_2), \ldots, (u_{t-1}, r_{t-1}), u_t\}$.

3.2 Prompt Chaining

To enable seamless generalization across domains without any additional example dialogs in each target domain, RealTOD employs a two-stage prompt chaining mechanism, which consists of two sequential prompting phases: (*i*) example dialog generation that transforms an example dialog from a source domain into a target domain while maintaining task-specific consistency; and (*ii*) task adaptation that leverages the generated example dialog for in-context learning in the target domain.

Example Dialog Generation. The first phase constructs an example dialog in the target domain by leveraging the schema mapping between the source domain and target domain. Formally, the inputs to LLM in this phase include the source domain schema Σ_{d_x}, an example dialog $\mathcal{T}_{d_x}$ in the source domain, an instruction prompt P_1 specifying the transformation process, and the target domain schema Σ_{d_y}. The output is a new example dialog $\mathcal{T}_{d_y}$ that aligns with the intents $\mathcal{I}_{d_x}$ and associated slots $\mathcal{S}_{i_x}$ in the target domain d_y.

Task Adaptation. Once the example dialog $\mathcal{T}_{d_y}$ in target domain d_y is generated, the second phase leverages this as an in-context learning example to enhance the model's adaptation in the target domain. At each dialog turn t, the inputs to the LLM include the target domain schema Σ_{d_y}, the generated example dialog $\mathcal{T}_{d_y}$, the dialog history up to turn t (denoted as H_t), and an instruction prompt P_2 that guides the response generation process.

The LLM then produces the system response r_t, which can be either a natural language reply or an API call, depending on the current task context. Since a single dialog may span multiple domains, we can denote the set of target domains involved in a dialog session as $\{d_1, d_2, \ldots, d_m\} \subseteq D$, and extend to the formulation to condition on all relevant domain schemas $\{\Sigma_{d_j}\}_{j=1}^{m}$.

Instruction Prompt. The prompt P_1 begins with a task description on generating a dialog from a schema, then presents domain_X's schema and its sample conversation. It instructs the LLM to analyze this structure, apply it to domain_Y, and generate a corresponding conversation. (For the full prompt P_1, see Appendix B.) The instruction prompt P_2 consists of two main parts: a task description and general guidelines. It directs the system to collect required slot values before API calls and use search results for accurate responses. The guidelines emphasize limiting slot requests per turn and confirming user inputs before invoking the API call. (For the full prompt P_2, see Appendix B.)

3.3 Fine-Grained Feedback

Even SOTA LLMs can make errors when generating API calls. To minimize these errors and ensure successful API execution, RealTOD integrates a fine-grained feedback mechanism via a generic API parser. Given a domain schema Σ_{d_x} and an API call a, the parser verifies the correctness of the request before execution. If the API call conforms to the schema, it is passed for execution; otherwise, the parser provides fine-grained feedback to the LLM for correction. The verification process identifies three types of errors: (*i*) incorrect method name, where the API method does not match any intent $i \notin \mathcal{I}_d$; (*ii*) incorrect slot name, where a provided slot is not defined in the schema $s_m \notin \mathcal{S}_i$ for the given intent; and (*iii*) missing required slots, where required slots s_m with `is_required`(s_m) = True are absent in the API parameters. Upon detecting an error, the parser returns fine-grained feedback specifying the issue, allowing the LLM to refine its response.

4 User Simulator

Ideally, a TOD system should interact with real users in order to evaluate its effectiveness. However, engaging real users is often costly and time-consuming. To address this challenge, we develop a user simulator. An effective user simulator must first accurately convey its needs by specifying the required slot values (e.g., 'departure city") before optionally requesting information (e.g., the

flight's arrival time") from the TOD system. To construct such a simulator, we utilize dialog data $\mathcal{T}_{d_x}$ consisting of user goals, expressed through API calls $A = [a_1, a_2, \ldots, a_n]$, and the request slots $R = [s_1, s_2, \ldots, s_m]$ that the user should request. To train the user simulator, we optimize an instruction-finetuned model as:

$$\mathcal{L} = -\sum_{k=1}^{|u_t|} \log p(w_k \mid w_{<k}, H_t, A, R),$$

where w_k denotes the k-th token in the user utterance u_t at turn t, and $w_{<k}$ represents all preceding tokens in the same utterance. The simulator learns to express user goals in natural language by providing values for requested slots, and request information from the TOD system, conditioned on the set of API calls A, request slots R, and dialog context H_t. To conduct an interactive session between a trained user simulator and the TOD system, the simulator initiates the conversation by retrieving the first user goal a_1 from A and associated request slot s_1 from R. This process continues iteratively until all user goals in A and their associated request slots in R have been processed.

5 Experiments

5.1 Datasets

We conduct our experiments using two datasets: the Schema-Guided dialog (SGD) dataset (Rastogi et al., 2020) and the Bilingual Task-Oriented dialog (BiToD) dataset (Lin et al., 2021b). Since BiToD includes dialogs in both Chinese and English, we retain only the English dialogs for our analysis. Both datasets provide domain-specific schemas along with corresponding dialog conversations, which are essential for baseline models. A comparative summary of key statistics for both datasets is presented in Table 1.

5.2 Experimental Setup

We integrated four LLMs in RealTOD: two open-source models, DeepSeek-V3 (Liu et al., 2024) and Llama-3.3-70B-Instruct (Dubey et al., 2024), and two proprietary models, GPT-4o (Achiam et al., 2023) and Claude 3.5 Sonnet (Anthropic, 2023). For GPT-4o, we accessed the model via the official OpenAI API[2], while Claude 3.5 Sonnet was queried using the official Anthropic API[3].

[2] https://openai.com/api/

[3] https://docs.anthropic.com/claude

Statistic	SGD	Bitod
Total Dialogs	4,201	352
Total Dialogs (Single-domain)	1,331	111
Total Dialogs (Multi-domain)	2,870	241
Total API Calls	13,239	1,005
Total API Calls (Single-domain)	2,188	127
Total API Calls (Multi-domain)	11,051	878
Total Turns	89,428	6,979
Total User Req. Slots	8,271	500
Avg. API calls per dialog	3.15	2.85
Avg. API calls (Single-domain)	1.64	1.14
Avg. API calls (Multi-domain)	3.85	3.64
Avg. turns per dialog	21.28	19.82
Avg. User Req. Slots	1.96	1.42
Avg. parameters per API call	2.96	3.51
Total Unique API methods	34	7
Total Unique API parameters	88	20

Table 1: Test Dataset Statistics for SGD and BiTOD.

We fine-tune Flan-T5 model (Chung et al., 2024) to act as a user simulator for each dataset. Specifically, we use the "google/flan-t5-base" model, which consists of 250 million parameters. During fine-tuning, we set the warm-up steps to 100 and applied early stopping based on the evaluation loss, with patience of three. The models were trained for 10 epochs.

5.3 Evaluation Metrics

To comprehensively evaluate the performance of RealTOD and baseline models, we assess the following: (*i*) Dialog-Level System Response, (*ii*) Inform Accuracy, (*iii*) API Call, and (*iv*) Dialog Success Rate.

Dialog-Level System Response. To assess the quality of the responses generated by RealTOD, we removed all user responses produced by our user simulator, retaining only system responses. We then concatenated all system turns into a single text containing only system-generated outputs. The same process was applied to the ground truth dialog, keeping and concatenating only the system turns. Finally, we evaluated system response quality at the dialog level by comparing the generated responses to the ground truth using BERTScore (Zhang et al., 2019), a metric that measures semantic similarity between texts. Furthermore, we utilize "microsoft/mpnet-base" as the foundational model for computing BERTScore.

Inform Accuracy. To evaluate how effectively RealTOD informs the user about the requested slots, we implemented a regex-based system. First, we

identify the slots requested by the user and extract their corresponding values from the search results. Then, we use regex matching to determine whether the system's subsequent responses include those extracted values. If a system turn contains the requested slot values, we consider the system to have successfully provided the required information.

API Calls. To evaluate the quality of API Calls, we first extract the key-value pairs $(name(s_k), v_k)_{k=1}^{n}$, along with method name i from the generated API call using regular expressions. *Method Accuracy* evaluates whether the generated API call uses the correct method name, assessed using exact matching. *Parameter Name Accuracy* determines whether all ground truth key names are included in the generated API call, using fuzzy matching. *Parameter Value Accuracy* verifies if the value associated with a correctly predicted key matches the ground truth, also using fuzzy matching. Notably, this metric is computed only when the corresponding *Parameter Name* is correctly predicted. *Operator Accuracy* applies specifically to the BiToD dataset, where API calls include operators (e.g., "at_least", "one_of"). We assess this using fuzzy matching. *Full API Accuracy* measures whether the entire API call – including the method, parameter, values, and, for BiToD, the operator – matches the ground truth.

Dialog Success Rate. This metric measures the percentage of dialogs in which all API calls achieve 100% *Full API Accuracy*. In other words, it represents the proportion of dialogs where every generated API call matches the ground truth, ensuring complete correctness throughout the dialog.

5.4 Baseline Methods

We compare RealTOD against several strong baseline models.

SimpleTOD (Hosseini-Asl et al., 2020) treats task-oriented dialog as a single sequence generation problem, using a causal language model to predict dialog state, actions, and responses auto regressively.

SOLOIST (Peng et al., 2021) is a Transformer-based task-oriented dialog system that unifies multiple dialog modules into a single pre-trained model. It leverages transfer learning and machine teaching, allowing adaptation to new tasks with minimal labeled data.

ZS-TOD (Mosharrof et al., 2023) is a zero-shot task-oriented dialog system that generalizes to unseen domains using domain schemas instead of memorizing task-specific patterns. It replaces full dialog history with a concise summary (previous dialog state), reducing context complexity.

AutoTOD (Xu et al., 2024) is a zero-shot task-oriented dialog agent that eliminates traditional modules, relying only on instruction-following LLMs like GPT-4. It requires no task-specific training and autonomously decides actions, queries APIs, and generates responses.

6 Results and Analysis

6.1 Evaluating the Quality of API Calls

Table 2 presents the API call accuracy results on both the SGD and BiToD datasets.

Comparing RealTOD Performance to Baselines. A key observation is that across both datasets, nearly all variants of RealTOD outperform the baseline models across all evaluation metrics, including Method Accuracy, Param Names Accuracy, Param Values Accuracy, Operator Accuracy (for BiToD), and Full API Accuracy. Notably, when focusing on Full API Accuracy, we see substantial gains of RealTOD over baselines. For instance, Claude surpasses AutoTOD, the strongest baseline, by 37.74% in Full API Accuracy on the SGD dataset. Similarly, on BiToD, GPT-4o outperforms SimpleTOD, the best baseline model, by 11.26%, highlighting the robustness of our approach. Moreover, to view the dialogs generated by RealTOD, please refer to Appendix B.

Open-Source vs. Proprietary Models. Another notable trend in the Table 2 is the consistent superiority of proprietary models (GPT-4o, Claude) over open-source models (DeepSeek, Llama) in Full API Accuracy across both datasets. For example, on the SGD dataset, Claude achieves an 18.91% higher Full API Accuracy than Llama, highlighting the performance gap between proprietary and open-source LLMs.

Model-Specific Observations. Interestingly, when comparing Llama and DeepSeek in the Table 2, their relative performance depends on the dataset. While Llama yields higher accuracies in most metrics on SGD, the trend reverses in BiToD, where DeepSeek significantly outperforms Llama. We attribute this to DeepSeek's closer alignment with Chinese data, which proves advantageous for BiToD's English subset that still contains Chinese references (e.g., restaurant names). This shows that LLM performance in TOD tasks depends on alignment with the dataset's language and domain.

Dataset	LLM Model	Method Accuracy			Param Names Accuracy			Param Values Accuracy			Operator Accuracy			Full API Accuracy		
		Single	Multi	Both	Single	Multi	Both	Single	Multi	Both	Single	Multi	Both	Single	Multi	Both
SGD	SOLOIST	61.56	65.10	64.51	44.85	47.50	47.06	42.96	45.60	45.16				24.50	26.89	26.50
	SimpleTOD	53.52	59.46	58.48	44.44	50.07	49.14	41.97	47.35	46.46	N/A	N/A	N/A	17.05	21.86	21.07
	ZS-TOD	74.36	50.00	54.26	64.74	41.54	45.60	62.82	39.23	43.35				35.90	16.3	19.73
	AutoTOD	56.67	62.47	61.52	58.49	64.82	63.77	54.76	61.32	60.23				41.96	47.91	46.92
	RealTOD-GPT-4o	80.60	71.26	72.81	84.72	73.54	75.40	81.44	70.57	72.38				68.71	57.89	59.69
	RealTOD-Claude	**88.29**	**81.74**	**82.83**	**88.91**	**80.59**	**81.97**	**85.66**	**77.08**	**78.51**	N/A	N/A	N/A	**72.58**	**63.04**	**64.63**
	RealTOD-DeepSeek	81.11	69.25	71.22	83.78	70.80	72.96	79.42	66.42	68.58				63.27	51.67	53.59
	RealTOD-Llama	80.32	69.60	71.39	86.65	71.46	73.99	82.36	67.56	70.02				63.23	52.58	54.35
BiTOD	SOLOIST	39.39	63.48	60.95	21.06	57.51	53.69	21.06	57.01	53.24	20.15	54.69	51.07	15.15	46.10	42.86
	SimpleTOD	25.00	59.54	56.21	25.00	58.94	55.66	25.00	58.81	55.55	24.59	56.58	53.49	21.43	50.00	47.24
	ZS-TOD	23.08	33.45	32.59	23.08	32.46	31.68	20.83	30.98	30.14	23.08	31.68	30.96	15.38	19.86	19.49
	AutoTOD	64.29	48.42	49.84	41.73	23.84	25.44	38.45	21.84	23.33	31.48	21.15	22.08	17.86	14.04	14.38
	RealTOD-GPT-4o	82.81	73.82	74.97	82.81	71.20	72.69	82.31	**66.90**	68.88	79.56	70.55	71.71	68.75	**50.17**	**52.56**
	RealTOD-Claude	**94.49**	66.74	**83.07**	**91.47**	69.06	**79.15**	**90.05**	60.19	**73.34**	**90.81**	**76.38**	**78.22**	**71.65**	47.30	50.40
	RealTOD-DeepSeek	90.55	**75.92**	77.79	86.93	**72.02**	73.92	84.69	65.12	67.62	85.33	71.35	73.13	62.20	44.47	46.73
	RealTOD-Llama	85.16	61.78	64.81	83.46	60.84	63.77	82.43	54.16	57.82	79.27	59.19	61.78	66.41	35.88	39.83

Table 2: API Call Accuracy breakdown across all models on the SGD and BiTOD datasets. Accuracy is reported across multiple metrics, including method, parameter name, parameter value, operator, and overall full API accuracy. Results are shown for single-domain, multi-domain, and both domains.

6.2 Dialog-Level System Response

Dialog-Level System Response. From Table 3, we observe that fine-tuned models such as SOLOIST and SimpleTOD generally yield higher BERTScores than LLM-powered models (including our RealTOD and AutoTOD). This is most evident on the SGD dataset, where SOLOIST achieves the highest BERTScores and SimpleTOD likewise surpasses LLM-based methods. This suggests that supervised fine-tuning enables closer alignment with reference responses, as measured by semantic overlap (BERTScore).

Inform Accuracy. Despite lower BERTScores, LLM-powered models excel at Inform Accuracy, which measures whether the correct slot values are returned to the user. On SGD, our RealTOD approach consistently attains the highest Inform Accuracy across Single/Multi/Both domains. Notably, RealTOD demonstrates an 82.93% Inform Accuracy in Single-domain settings, substantially higher than SOLOIST (44.54%). Similarly, on BiToD, AutoTOD shows particularly strong Inform Accuracy (reaching over 90% in Single-domain settings), outperforming SOLOIST and SimpleTOD. These results confirm that LLMs, especially when guided by a dedicated system architecture (e.g., RealTOD), tend to be more precise in providing the requested slot values – even if their surface-level similarity to the reference text is lower.

6.3 Ablation Study

To assess the effectiveness of our proposed components – Fine-Grained Feedback and Prompt Chaining – we conducted an ablation study using 100 dialog conversations sampled from the SGD dataset (50 from multi-domain and 50 from single-domain). We evaluated all four variants of RealTOD (GPT-4o, Claude, Llama, and DeepSeek) under four different settings: one without either of the components, one with Fine-Grained Feedback only, one with Prompt Chaining only, and one with both components. Moreover, we used the Full API Accuracy as our comparison metric. This experimental design allowed us to isolate the impact of each component and determine their individual and combined contributions to performance. The results are provided in the Table 4. As it can be seen in Table 4, adding Fine-Grained Feedback alone leads to moderate improvements, indicating its role in refining APIs. Prompt Chaining, on the other hand, provides a more substantial boost. The combination of both components yields the highest accuracy, demonstrating their complementary nature.

Dataset	LLM Model	BERTScore System (F1)			Inform Accuracy		
		Single	Multi	Both	Single	Multi	Both
SGD	SOLOIST	**0.7132**	**0.7096**	**0.7107**	44.54	54.42	52.66
	SimpleTOD	0.6753	0.6758	0.6756	32.15	51.11	47.74
	ZS-TOD	0.5119	0.5139	0.5136	12.32	11.20	11.40
	AutoTOD	0.5716	0.5937	0.5867	76.34	75.84	75.93
	RealTOD-GPT-4o	0.6547	0.6544	0.6545	**82.93**	**76.89**	**78.03**
	RealTOD-Claude	0.6552	0.6694	0.6649	79.44	71.37	72.85
	RealTOD-DeepSeek	0.6345	0.6384	0.6372	82.81	75.45	76.88
	RealTOD-Llama	0.6019	0.5979	0.5992	73.49	70.26	70.88
BiTOD	SOLOIST	0.5479	0.6977	0.6572	80.0	69.42	70.62
	SimpleTOD	0.5292	**0.7103**	**0.6636**	85.0	71.33	72.88
	ZS-TOD	0.5729	0.655	0.6319	60.0	66.24	65.53
	AutoTOD	0.5064	0.5358	0.5277	**97.50**	**84.39**	**85.87**
	RealTOD-GPT-4o	**0.6543**	0.6447	0.6477	85.18	79.78	80.26
	RealTOD-Claude	0.6523	0.6392	0.6434	64.70	63.51	63.61
	RealTOD-DeepSeek	0.6454	0.6278	0.6334	86.66	79.25	79.93
	RealTOD-Llama	0.5927	0.5694	0.5769	64.28	65.06	65.00

Table 3: Comparison of RealTOD with baseline models for inform accuracy and BERTScore for system response on SGD and BiTOD datasets.

Fine-Grained Feedback	Prompt Chaining	GPT-4o	Claude	DeepSeek	Llama
✗	✗	51.89	29.92	45.83	49.62
✓	✗	56.06	36.74	48.10	57.19
✗	✓	64.01	70.45	54.92	59.84
✓	✓	**66.66**	**72.34**	**59.46**	**63.63**

Table 4: Ablation study of full API accuracy on the SGD dataset to evaluate the impact of fine-grained feedback and prompt chaining across different LLMs.

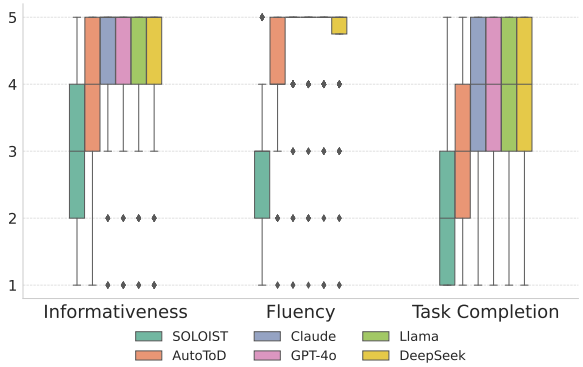

Figure 2: Human evaluation results on SGD and BiTOD. Human evaluators were asked to rate the generated conversations on a scale of 1 to 5 across three key aspects: informativeness, naturalness, and task completion.

6.4 Human Evaluation

We conducted a human evaluation using Amazon Mechanical Turk to assess the performance of our models. We used two baseline models to compare with all four variants of RealTOD. For our evaluation, we sampled 100 dialogs from the test sets of our chosen datasets (SGD and BiToD), with 50 each from single and multi-domain tasks. We asked the human evaluators to rate the generated conversations on a scale of 1 to 5 across three key aspects: *Informativeness*, *Naturalness*, and *Task Completion Rate*. Figure 2 shows the human evaluation results, where all four variants of RealTOD outperformed the baseline models (SOLOIST and AutoToD), supporting the reliability of our evaluation metrics. Moreover, the Claude variant of RealTOD achieved the highest average scores across all three aspects, albeit by a slight margin, further verifying the accuracy of our metrics.

6.5 Dialog Success Rate

To rigorously assess the quality of the generated dialogs, we conducted an experiment to measure the dialog success rate as the number of API calls within a dialog increases. As shown in Figure 3, all variants of RealTOD exhibit a declining trend in dialog success rate as the number of API calls

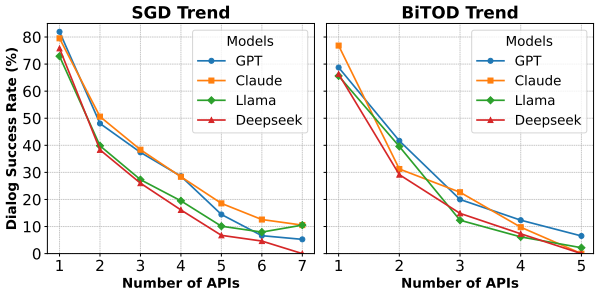

Figure 3: Trend in dialog success rate across models: we notice a decline in dialog success rate as the number of API calls increases for different models across the SGD and BiTOD datasets. This trend highlights the challenge of error propagation in LLM-powered TOD systems, where mistakes in the earlier part of the dialog negatively impact subsequent interactions.

increases. This trend is consistent across both the SGD and BiTOD datasets. The primary reason for this decline is the interdependence of API calls. For example, when a user books a restaurant at a particular destination, the same location is often referenced for booking a taxi or later searching for nearby hotels. Any errors in earlier API calls can propagate, making subsequent calls more prone to failure. The Figure 3 highlights a limitation of LLM-powered TOD systems, suggesting that they are still far from achieving perfect performance. Further research is needed to enhance their ability to handle these scenarios more effectively.

7 Conclusion

We introduce RealTOD, a novel framework that enhances LLM-based TOD systems through prompt chaining and fine-grained feedback. RealTOD enables zero-shot domain adaptation by automatically generating in-domain demonstrations, while its fine-grained feedback mechanism systematically verifies API calls and provides precise corrective actions. Our approach significantly improves multi-turn task completion without domain-specific fine-tuning. Our experiments on SGD and BiTOD datasets demonstrate that RealTOD achieves substantial gains in Full API Call Accuracy, surpassing state-of-the-art TOD systems. Human evaluations further confirm that LLMs integrated with RealTOD generate more fluent, informative, and effective task completions than baseline models. Ablation studies highlight the complementary contributions of prompt chaining and fine-grained feedback. RealTOD paves the way for more scalable and adaptable TOD systems by eliminating the need for domain-specific fine-tuning.

8 Limitations

We restricted our experiments to four popular LLMs (GPT-4o, Claude 3.5 Sonnet, DeepSeek-V3, and Llama-3.3-70B-Instruct) due to time and computational constraints. Given the rapid pace of model development – exemplified by emerging systems such as DeepSeek R1 (Guo et al., 2025) and OpenAI o3-mini (OpenAI), Qwen-2.5 Max (Team, 2024) – it remains an open question how RealTOD would perform on these newer LLMs.

While the user simulators fine-tuned in this study perform reasonably well, they are not perfect and may occasionally struggle to respond accurately to RealTOD. Further details on their performance can be found in Appendix A.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2023. Claude. https://www.anthropic.com/claude. Accessed: 2024-03-01.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wenhu 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*, pages 3696–3709, Florence, Italy. Association for Computational Linguistics.

Qinyuan Cheng, Linyang Li, Guofeng Quan, Feng Gao, Xiaofeng Mou, and Xipeng Qiu. 2022. Is MultiWOZ a solved task? an interactive TOD evaluation framework with user simulator. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1248–1259, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Willy Chung, Samuel Cahyawijaya, Bryan Wilie, Holy Lovenia, and Pascale Fung. 2023. InstructTODS: Large language models for end-to-end task-oriented dialogue systems. In *Proceedings of the Second Workshop on Natural Language Interfaces*, pages 1–21, Bali, Indonesia. Association for Computational Linguistics.

Sam Davidson, Salvatore Romeo, Raphael Shu, James Gung, Arshit Gupta, Saab Mansour, and Yi Zhang. 2023. User simulation with large language models for evaluating task-oriented dialogue. *arXiv preprint arXiv:2309.13233*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

W. Eckert, E. Levin, and R. Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87.

Moghis Fereidouni, Adib Mosharrof, and A.b. Siddique. 2024. Grounded language agent for product search via intelligent web interactions. In *Proceedings of the 1st Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*, pages 63–75, Miami, Florida, USA. Association for Computational Linguistics.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2024. Multimodal web navigation with instruction-finetuned foundation models. In *The Twelfth International Conference on Learning Representations*.

Jun Gao, Liuyu Xiang, Huijia Wu, Han Zhao, Yiqi Tong, and Zhaofeng He. 2023. An adaptive prompt generation framework for task-oriented dialogue system. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1078–1089, Singapore. Association for Computational Linguistics.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Wanwei He, Yinpei Dai, Binyuan Hui, Min Yang, Zheng Cao, Jianbo Dong, Fei Huang, Luo Si, and Yongbin Li. 2022. SPACE-2: Tree-structured semi-supervised contrastive pre-training for task-oriented dialog understanding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 553–569, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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, 33:20179–20191.

Vojtěch Hudeček and Ondrej Dusek. 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*, pages 216–228, Prague, Czechia. Association for Computational Linguistics.

Wiradee Imrattanatrai and Ken Fukuda. 2023. End-to-end task-oriented dialogue systems based on schema. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10148–10161, Toronto, Canada. Association for Computational Linguistics.

Nihal Jain, Robert Kwiatkowski, Baishakhi Ray, Murali Krishna Ramanathan, and Varun Kumar. 2024. On mitigating code llm hallucinations with api documentation. *arXiv preprint arXiv:2407.09726*.

Byoungjip Kim, Youngsoo Jang, Lajanugen Logeswaran, Geon-Hyeong Kim, Yu Jin Kim, Honglak Lee, and Moontae Lee. 2024. Prospector: Improving LLM agents with self-asking and trajectory ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14958–14976, Miami, Florida, USA. Association for Computational Linguistics.

Tiziano Labruna, Sofia Brenna, Andrea Zaninello, and Bernardo Magnini. 2023. Unraveling chatgpt: A critical analysis of ai-generated goal-oriented dialogues and annotations. In *AI*IA*, pages 151–171.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484.

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 442–451, Metz, France. Association for Computational Linguistics.

Sunjae Lee, Junyoung Choi, Jungjae Lee, Munim Hasan Wasi, Hojun Choi, Steven Y Ko, Sangeun Oh, and Insik Shin. 2023. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. *arXiv preprint arXiv:2312.03003*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Melbourne, Australia. Association for Computational Linguistics.

Hsien-chin Lin, Christian Geishauser, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck, and Milica Gasic. 2022. GenTUS: Simulating user behaviour and language in task-oriented dialogues with generative transformers. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 270–282, Edinburgh, UK. Association for Computational Linguistics.

Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gasic. 2021a. Domain-independent user simulation with transformers for task-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 445–456, Singapore and Online. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. MinTL: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405, Online. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, Peng Xu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021b. Bitod: A bilingual multi-domain dataset for task-oriented dialogue modeling. *arXiv preprint arXiv:2106.02787*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Hong Liu, Yucheng Cai, Zhijian Ou, Yi Huang, and Junlan Feng. 2022. A generative user simulator with GPT-based architecture and goal state tracking for reinforced multi-domain dialog systems. In *Proceedings of the Towards Semi-Supervised and Reinforced Task-Oriented Dialog Systems (SereTOD)*, pages 85–97, Abu Dhabi, Beijing (Hybrid). Association for Computational Linguistics.

Kaixin Ma, Hongming Zhang, Hongwei Wang, Xiaoman Pan, and Dong Yu. 2023. LASER: LLM agent with state-space exploration for web navigation. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2Seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478, Melbourne, Australia. Association for Computational Linguistics.

Fei Mi, Yasheng Wang, and Yitong Li. 2022. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11076–11084.

Adib Mosharrof, Muhammad Hasan Maqbool, and AB Siddique. 2023. Zero-shot generalizable end-to-end task-oriented dialog system using context

summarization and domain schema. *arXiv preprint arXiv:2303.16252*.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

OpenAI. o3. [Large Language Model].

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*, 9:807–824.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. 2023. Summarization is (almost) dead. *ArXiv*, abs/2309.09558.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Lei Shu, Liangchen Luo, Jayakumar Hoskere, Yun Zhu, Yinxiao Liu, Simon Tong, Jindong Chen, and Lei Meng. 2024. Rewritelm: An instruction-tuned large language model for text rewriting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18970–18980.

A.B. Siddique, M.H. Maqbool, Kshitija Taywade, and Hassan Foroosh. 2022. Personalizing task-oriented dialog systems via zero-shot generalizable reward function. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1787–1797, New York, NY, USA. Association for Computing Machinery.

Yewei Song, Cedric Lothritz, Xunzhu Tang, Saad Ezzini, Jacques Klein, Tegawendé F Bissyandé, Andrey Boytsov, Ulrick Ble, and Anne Goujon. 2025. Callnavi: A study and challenge on function calling routing and invocation in large language models. *arXiv preprint arXiv:2501.05255*.

Abishek Sridhar, Robert Lo, Frank F. Xu, Hao Zhu, and Shuyan Zhou. 2023. Hierarchical prompting assists large language model on web navigation. In *Conference on Empirical Methods in Natural Language Processing*.

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)*, pages 4661–4676, Dublin, Ireland. Association for Computational Linguistics.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8990–9005, Singapore. Association for Computational Linguistics.

Qwen Team. 2024. Qwen 2.5 max: A large language model for advanced natural language processing. `https://www.example.com/qwen-max`. Accessed: 2024-03-01.

Silvia Terragni, Modestas Filipavicius, Nghia Khau, Bruna Guedes, André Manso, and Roland Mathis. 2023. In-context learning user simulators for task-oriented dialog systems. *arXiv preprint arXiv:2306.00774*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam M. Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, Yaguang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, I. A. Krivokon, Willard James Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Hartz Søraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Díaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, V. O. Kuzmina, Joseph Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Rogers Croak, Ed H. Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *ArXiv*, abs/2201.08239.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable dialogue systems and user simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1:*

Long Papers), pages 152–166, Online. Association for Computational Linguistics.

Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerova, et al. 2023. Clinical text summarization: adapting large language models can outperform human experts. *Research Square*.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024b. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*.

Zhiqiang Wang, Yiran Pang, and Yanbin Lin. 2023. Large language models are zero-shot text classifiers. *arXiv preprint arXiv:2312.01044*.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, page 543–557, New York, NY, USA. Association for Computing Machinery.

Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. 2023. Droidbot-gpt: Gpt-powered ui automation for android. *arXiv preprint arXiv:2304.07061*.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and Heyan 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)*, pages 2748–2763, Bangkok, Thailand. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024a. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Yazhou Zhang, Mengyao Wang, Chenyu Ren, Qiuchi Li, Prayag Tiwari, Benyou Wang, and Jing Qin. 2024b. Pushing the limit of llm capacity for text classification. *arXiv preprint arXiv:2402.07470*.

A User Simulator Performance

Dataset	LLM Model	BERTScore User (F1)			BERTScore Overall (F1)		
		Single	Multi	Both	Single	Multi	Both
SGD	RealTOD-GPT-4o	0.6879	0.6712	0.6765	0.6821	0.6761	0.6780
	RealTOD-Claude	0.7053	0.6972	0.6997	0.6822	0.6852	0.6843
	RealTOD-DeepSeek	0.6844	0.6700	0.6746	0.6600	0.6554	0.6569
	RealTOD-Llama	0.6853	0.6766	0.6794	0.6365	0.6256	0.6291
BiTOD	RealTOD-GPT-4o	0.6794	0.6549	0.6627	0.6889	0.6671	0.6741
	RealTOD-Claude	0.6963	0.6737	0.6809	0.6841	0.6625	0.6693
	RealTOD-DeepSeek	0.6979	0.6502	0.6653	0.6789	0.6530	0.6612
	RealTOD-Llama	0.6534	0.6370	0.6423	0.6345	0.5990	0.6104

Table 5: Performance of the Dialog-Level User Simulator and Entire Conversation.

To evaluate the performance of our user simulator, we computed BERTScore on user turns (BERTScore User) and on the concatenation of both user and system turns (BERTScore Overall). As shown in Table 5, most BERTScores fall within the range of approximately 0.6 to 0.7, showing that the simulated user responses maintain a reasonable level of similarity to the reference.

B Example Dialog Responses

Tables 6, 7, 8, and 9 present an example of a multi-domain conversation where our User Simulator interacts with different variants of RealTOD (Claude, GPT-4o, DeepSeek, and Llama) to check the weather, schedule a property visit, and reserve a car. Additionally, Tables 10 and 11 showcase examples generated by the two baseline models SOLOIST and ZS-TOD. Both of these models struggle with task completion; SOLOIST, on the second turn, missed the API call. The same thing happened when ZS-TOD completely forgot to make the API call when handling a car reservation, ultimately failing to provide the requested information. For model errors, we use ✗, and for correct API calls, we use ✓.

Instruction Prompt Template P_1

Task Description:
Your task is to generate a dialog conversation between a User and a System based on a given domain schema. I will provide a **Schema** for **{domain_X}**, which defines the structure and relevant entities, along with a corresponding dialog conversation for reference. Your goal is to analyze the relationship between the dialog and the schema and then generate a coherent and contextually appropriate dialog conversation for **{domain_Y}** while maintaining consistency with its schema.

Here is a **Schema** for the **{domain_X}**
service_name: {domain_X}
Intents
intent_no.
name: IntentName
is_transactional: True/False
required_slots: required slot 1, required slot 2, required slot 3, ...
optional_slots: optional slot 2, optional slot 2, optional slot 3, ...
Slots
slot_name: slot name 1, slot name 2, slot name 3, ...
possible_values: value 1, value 2, value 3, ...
end of schema for {domain_X}
A sample **« `Dialog Conversation` »** between a System and a User will be fetched.

Now, understand the above conversation structure between a User and a System. You will be given a new **Schema** for **{domain_Y}**. You have to generate a full-fledged conversation for the new domain that will be structured like the example above.

Here is a **Schema** for the **{domain_Y}**
service_name: {domain_Y}
Intents
intent_no.
name: IntentName
is_transactional: True/False
required_slots: required slot 1, required slot 2, required slot 3, ...
optional_slots: optional slot 2, optional slot 2, optional slot 3, ...
Slots
slot_name: slot name 1, slot name 2, slot name 3, ...
possible_values: value 1, value 2, value 3, ...
end of schema for {domain_Y}

Based on the above instructions and example conversation from the domain_X, learn how to generate the full conversation for the new domain_Y domain.
End of Instructions.

Instruction Prompt Template P_2

Task Description:
Think of yourself as an expert chat assistant specialized in the {domain_name} domain. Your task is to generate the most natural and helpful responses for a given task-oriented dialog context. I will provide **Schema** for {domain_name}, one sample conversation between a System and a User, optionally, search results from the database. Understand the dialog relation to **Schema**. You can request slot values from the User to fulfill the User's current **intent**. Remember that required **slots** are more important than optional **slots**. When making **API** calls, use column names from the **Schema** as parameters. Match the required and optional **slots** with the column names and use them in **API** calls. Before making the call, ensure you've gathered all required **slots** from the User. You can skip unnecessary parameters.

Here is a **Schema** for the domain_name
service_name: {domain_name}
Intents
intent_no.
name: IntentName
is_transactional: True/False
required_slots: required slot 1, required slot 2, required slot 3, ...
optional_slots: optional slot 2, optional slot 2, optional slot 3, ...
Slots
slot_name: slot name 1, slot name 2, slot name 3, ...
possible_values: value 1, value 2, value 3, ...
end of schema
A sample **« `Dialog Conversation` »** between a System and a User will be fetched.

Understand the above structure of conversation between a User and a System. Learn how to interact with the User and generate the most human-like conversational response to the User's **intent**. You may need to make **API** Calls and use the **API** Call results. Based on the above instructions and examples from the {domain_name} domain, learn how to interact with a User to generate the most human-like conversational response to the User's current **intent**.
End of Instructions.

Here are a few general **Guidelines** to follow:
- Please avoid asking for too many **slots** in one turn; ideally, ask one slot at a time.
- Don't overwhelm the User with too many questions or choices in one turn.
- Confirm the slot values with the User before finalizing the **API** Call.
- Follow the structure of **API** Call from the above example whenever you are making an **API** Call.
- If you're unsure about something, it's always better to ask or confirm with the User.
- Do not provide all the information in the search results to the User. Provide details only if the User requests them.
- If you feel the User is confused, guide the User with relevant suggestions and ensure it is relevant to their current **intent**.
- You generate only one system response at a time and do not produce search results yourself; search results will be provided to you.

Conversation history: « conversation history » up to turn t will be fetched

Table 6: Multi-Domain SGD Conversation - RealTOD (Claude 3.5 Sonnet)

SGD Multi-Domain Conversation (Claude 3.5 Sonnet)
Goal: User is trying to check the weather in Vancouver, then schedule a property visit at the Golf Club Manor Apartments, and reserve a Hatchback car
User: What's the weather going to be like in Vancouver, BC on March 2nd?
RealTOD: I'll help you check the weather in Vancouver, BC for March 2nd. APICall(method='GetWeather', parameters={ city: Vancouver, date: 2024-03-02 }) ✓
Search Results: [{'city': 'Vancouver', 'date': '2024-03-02', 'temperature': 68, 'precipitation': 25, 'humidity': 26, 'wind': 6}]
RealTOD: On March 2nd in Vancouver, the temperature will be 68 degrees Fahrenheit with a 25 percent chance of precipitation. The humidity will be at 26% with wind speeds of 6 mph. **User:** Okay, sounds good. I'd like to schedule a visit to the Golf Club Manor Apartments. **RealTOD:** I can help you schedule a property visit to Golf Club Manor Apartments. I notice you were checking the weather for March 2nd. Would you like to schedule the visit for that same date? **User:** Yes, I'd like to schedule the visit.
RealTOD: APICall(method='ScheduleVisit', parameters={ property_name: Golf Club Manor Apartments, visit_date: 2024-03-02 }) ✓
Search Results: [{'property_name': 'Golf Club Manor Apartments', 'address': '375 Industrial Parkway # 314', 'area': 'Hayward', 'price': 2500000, 'phone_number': '5105810911'}]
RealTOD: I've confirmed your visit to Golf Club Manor Apartments for March 2nd. The property is located at 375 Industrial Parkway # 314 in Hayward. Would you like to know any specific details about the property? **User:** Yes, I'd like to reserve a car there. I'd like to pick it up at Indira Gandhi International Airport at 3 o'clock in the afternoon. I'd like a Hatchback. **RealTOD:** I'll help you reserve a Hatchback at Indira Gandhi International Airport. Would you like to pick up the car on the same date at 3:00 PM? Also, how long will you keep the car? **User:** Yes, that sounds good. I'll return the car on the 3rd of this month. **RealTOD:** I'll help you reserve a Hatchback from March 2nd to March 3rd. Would you like to add insurance to your rental? **User:** Yes, I'd like to add insurance.
RealTOD: APICall(method='ReserveCar', parameters={ pickup_location: Indira Gandhi International Airport, car_type: Hatchback, start_date: 2019-03-02, end_date: 2019-03-03, pickup_time: 15:00, add_insurance: True }) ✓
Search Results: [{'car_name': 'Fiat Panda', 'car_type': 'Hatchback', 'pickup_location': 'Indira Gandhi International Airport', 'pickup_time': '15:00', 'price_per_day': 39.0, 'add_insurance': True}]
RealTOD: Your car reservation has been confirmed. You'll be getting a Fiat Panda Hatchback, which will cost $39.00 per day with insurance included. You can pick it up at Indira Gandhi International Airport on March 2nd at 3:00 PM. Would you like to know any additional details about your car rental? **User:** No, thank you for your help. **RealTOD:** I'm happy to assist. Have a great day!

Table 7: Multi-SGD Conversation - RealTOD (GPT-4o)

SGD Multi-Domain Conversation (GPT-4o)
Goal: User is trying to check the weather in Vancouver, then schedule a property visit at the Golf Club Manor Apartments, and reserve a Hatchback car
User: What's the weather going to be like in Vancouver, BC on March 2nd?
RealTOD: APICall(method='GetWeather', parameters={ city: Vancouver, date: 2019-03-02 }) ✓
Search Results: [{'city': 'Vancouver', 'date': '2019-03-02', 'temperature': 67, 'precipitation': 49, 'humidity': 45, 'wind': 15}]
RealTOD: The average temperature for the day will be about 67 degrees Fahrenheit, and there's a 49 percent chance of rain. **User:** That sounds good. I'd like to schedule a visit to Golf Club Manor Apartments. **RealTOD:** Could you confirm that you want to visit Golf Club Manor Apartments on March 2nd? **User:** Sounds good.
RealTOD: APICall(method='ScheduleVisit', parameters={ property_name: Golf Club Manor Apartments, visit_date: 2019-03-02 }) ✓
Search Results: [{'property_name': 'Golf Club Manor Apartments', 'address': '375 Industrial Parkway # 314', 'number_of_baths': 3, 'number_of_beds': 3, 'phone_number': '510-581-0911', 'price': 3950}]
RealTOD: Your visit has been scheduled. **User:** I would like to reserve a car there till the 3rd. **RealTOD:** Do you like Hatchback, SUV, or another type of car? Would you like to pick up the car from March 2nd and from where? **User:** Sounds good. Delhi Airport is a good place for me to pick it up. A hatchback sounds perfect. **RealTOD:** Would you like insurance? What time will you pick it up? **User:** I'd like insurance and will pick it up at 3:00 PM. **RealTOD:** So you want to pick up the car from Indira Gandhi International Airport on March 2nd at 3 PM and drop it off on March 3rd? It's a Hatchback that comes with insurance. **User:** Sounds perfect.
RealTOD: APICall(method='ReserveCar', parameters={ add_insurance: True, car_type: Hatchback, end_date: 2019-03-03, pickup_location: Indira Gandhi International Airport, pickup_time: 15:00, start_date: 2019-03-02 }) ✓
Search Results: [{'car_name': 'Hyundai i20', 'car_type': 'Hatchback', 'pickup_location': 'Indira Gandhi International Airport', 'pickup_time': '15:00', 'price_per_day': 35.00, 'add_insurance': True}]
RealTOD: Your reservation for a Hyundai i20 Hatchback has been confirmed. The cost is $35.00 per day. If you have any more questions or need further assistance, feel free to ask! **User:** Thank you. That's all I need for now. **RealTOD:** Have a great day!

Table 8: Multi-SGD Conversation - RealTOD (DeepSeek-V3)

SGD Multi-Domain Conversation (DeepSeek-V3)
Goal: User is trying to check the weather in Vancouver, then schedule a property visit at the Golf Club Manor Apartments, and reserve a Hatchback car
User: What's the weather going to be like in Vancouver, BC on March 2nd?
RealTOD: APICall(method='GetWeather', parameters={ city: Vancouver, date: 2019-03-02 }) ✓
Search Results: [{'city': 'Vancouver', 'date': '2019-03-02', 'temperature': 45, 'precipitation': 10, 'humidity': 65, 'wind': 12}]
RealTOD: On March 2nd in Vancouver, BC, the temperature will be 45 degrees Fahrenheit with a 10 percent chance of rain. The wind speed will be 12 mph. **User:** I'd like to schedule a visit to the Golf Club Manor Apartments. **RealTOD:** Got it! I can help you schedule a visit to the Golf Club Manor Apartments. Could you please confirm the date you'd like to schedule the visit for? **User:** I'd like to visit the property on the 2nd of this month.
RealTOD: APICall(method='ScheduleVisit', parameters={ property_name: Golf Club Manor Apartments, visit_date: 2019-03-02 }) ✓
Search Results: [{'property_name': 'Golf Club Manor Apartments', 'address': '123 Fairway Drive', 'number_of_baths': 2, 'number_of_beds': 3, 'phone_number': '604-555-1234', 'price': 2500}]
RealTOD: Your visit to the Golf Club Manor Apartments on March 2nd has been confirmed. The property is located at 123 Fairway Drive and has 3 bedrooms and 2 bathrooms. Let me know if you'd like any additional details! **User:** I'd like to book a car to get there. I'd like to pick it up at Indira Gandhi International Airport, and I'd like it to be a Hatchback. **RealTOD:** I'll help you book a Hatchback for your trip. Could you please provide the date and time you'd like to pick up the car, the return date, and whether you'd like to add insurance? **User:** I'd like to pick up the car at 3 o'clock in the afternoon, and I'd like to return it on the 3rd of this month. I'd like to add insurance to my rental.
RealTOD: APICall(method='ReserveCar', parameters={ car_type: Hatchback, pickup_location: Indira Gandhi International Airport, start_date: 2019-03-02, pickup_time: 15:00, end_date: 2019-03-03, add_insurance: True }) ✓
Search Results: [{'car_name': 'Fiat Panda', 'car_type': 'Hatchback', 'pickup_location': 'Indira Gandhi International Airport', 'pickup_time': '15:00', 'price_per_day': 39.0, 'add_insurance': True}]
RealTOD: Your car reservation has been successfully made! You'll be getting a Fiat Panda Hatchback, which will cost $39.00 per day with insurance included. You can pick it up at Indira Gandhi International Airport on March 2nd at 3:00 PM. Let me know if you'd like any additional details! **User:** Thank you for your assistance. That will be all for now. **RealTOD:** I'm happy to assist. Have a great day!

Table 9: Multi-SGD Conversation - RealTOD (Llama-3.3-70B-Instruct)

SGD Multi-Domain Conversation (Llama-3.3-70B-Instruct)

Goal: User is trying to check the weather in Vancouver, then schedule a property visit at the Golf Club Manor Apartments, and reserve a Hatchback car

User: What's the weather going to be like in Vancouver, BC on March 2nd?
RealTOD: To provide you with the weather information for Vancouver, BC on March 2nd, I'll need to make an API call. Before I do that, I just want to confirm the details with you. You're looking for the weather in Vancouver, BC, and the specific date you're interested in is March 2nd, correct?
User: Yes, that's correct.

RealTOD: APICall(method='GetWeather', parameters={ city: Vancouver, date: 2019-03-02 }) ✓

Search Results: [{'city': 'Vancouver', 'date': '2019-03-02', 'temperature': 68, 'precipitation': 25, 'humidity': 26, 'wind': 6}]

RealTOD: It seems like the search results provided multiple weather forecasts for Vancouver, BC on March 2nd. To provide a clear answer, could you specify if you're more interested in temperature, precipitation, or humidity?
User: Yes, that's what I'm looking for. I'd like to schedule a visit to the Golf Club Manor Apartments on that day.
RealTOD: To schedule a visit to the Golf Club Manor Apartments, I need to confirm the details with you. You'd like to visit on March 2nd, correct?
User: Yes, that's correct.

RealTOD: APICall(method='ScheduleVisit', parameters={ property_name: Golf Club Manor Apartments, visit_date: 2019-03-02 }) ✓

Search Results: [{'property_name': 'Golf Club Manor Apartments', 'address': '375 Industrial Parkway # 314', 'number_of_baths': 1, 'number_of_beds': 2, 'phone_number': '5105810911', 'price': 2500000}]

RealTOD: Your visit to Golf Club Manor Apartments has been scheduled. The property is located at 375 Industrial Parkway # 314.
User: I'd like to reserve a car there and pick it up at Indira Gandhi International Airport.
RealTOD: Would you like a Sedan, Hatchback, or SUV? Also, what time would you like to pick it up?
User: I'd like a Hatchback and will pick it up at 3:00 PM.

RealTOD: APICall(method='ReserveCar', parameters={ add_insurance: True, car_type: Hatchback, end_date: 2019-03-03, pickup_location: Indira Gandhi International Airport, pickup_time: 15:00, start_date: 2019-03-02 }) ✓

Search Results: [{'car_name': 'Fiat Panda', 'car_type': 'Hatchback', 'pickup_location': 'Indira Gandhi International Airport', 'pickup_time': '15:00', 'price_per_day': 39.00, 'add_insurance': True}]

RealTOD: Your reservation for a Fiat Panda Hatchback has been confirmed. The cost is $39.00 per day.
User: Thank you very much. You've been very helpful. That will be all for now.
RealTOD: Have a great day!

Table 10: Multi-SGD Conversation Baseline Model Dialog Example -(SOLOIST)

SGD Multi-Domain Conversation (SOLOIST)
Goal: User is trying to check the weather in Vancouver, then schedule a property visit at the Golf Club Manor Apartments, and reserve a Hatchback car
User: I'd like to find out what the weather is going to be like on 2nd of March in Vancouver, BC.
SOLOIST: APICall: (Missed API Call Here) ✗
Search Results: [Missed Search Results] ✗
SOLOIST: The average temperature for the day should be 67 degrees Fahrenheit with a 49 percent chance of rain. **User:** That sounds good. I'd like to schedule a visit to Golf Club Manor Apartments. **SOLOIST:** What time do you want to pick it up? **User:** Sounds good.
SOLOIST: APICall(method='ScheduleVisit', parameters={ property_name: Golf Club Manor Apartments, visit_date: 2019-03-02nd }) ✓
Search Results: [{'address': '375 Industrial Parkway # 314', 'has_garage': False, 'in_unit_laundry': True, 'number_of_baths': 3, 'number_of_beds': 3, 'phone_number': '510-581-0911', 'price': 3950, 'property_name': 'Golf Club Manor Apartments', 'visit_date': '2019-03-02'}]
SOLOIST: Your visit has been scheduled. **User:** I would like to reserve a car there till the 3rd. **SOLOIST:** Could I help you with anything else? **User:** Sounds good. Delhi Airport is a good place for me to pick it up. A hatchback sounds perfect. **SOLOIST:** Please confirm: You want me to reserve a standard car from YVR International Airport on March 2nd at 3:30 pm until March 11th. **User:** I'd like insurance and will pick it up at 3 PM. **SOLOIST:** Do you want to pick it up on March 1st? You'll be picking it up at 3:30 pm. Is that right? **User:** Sounds perfect.
SOLOIST: APICall(method='GetCarsAvailable', parameters={ add_insurance: True, car_type: Standard, end_date: 2019-03-03, pickup_location: YVR International Airport, pickup_time: 15:30, start_date: 2019-03-01 }) ✗
Search Results: [{'car_name': 'Hyundai Elantra', 'car_type': 'Standard', 'pickup_location': 'VVR International Airport', 'pickup_time': '15:30', 'price_per_day': 34.0, 'add_insurance': True}]
SOLOIST: Your car has been reserved. **SOLOIST:** The car has been reserved. It is a Hyundai Elantra and it is $34. **User:** What is the name of the car and how much is it per day? **SOLOIST:** It is $34.00 per day and it's called a Hyundai Elantra. **User:** Thanks for helping. That's it for now. **SOLOIST:** Have a great day.

Table 11: Multi-SGD Conversation Baseline Model Dialog Example - (ZS-TOD)

SGD Multi-Domain Conversation (ZS-TOD)
Goal: User is trying to check the weather in Vancouver, then schedule a property visit at the Golf Club Manor Apartments, and reserve a Hatchback car
User: I'd like to find out what the weather is going to be like on 2nd of March in Vancouver, BC.
ZS-TOD: APICall(method='GetWeather', parameters={ city: Vancouver, date: 2019-03-02 }) ✓
Search Results: [city: Vancouver, date: 2019-03-02, humidity: 45, precipitation: 49, temperature: 67, wind: 15]
ZS-TOD: The average temperature for the day will be about 67 degrees Fahrenheit and there's a 49 percent chance of rain. **User:** That sounds good. I'd like to schedule a visit to Golf Club Manor Apartments. **ZS-TOD:** Could you confirm that you want to visit Golf Club Manor Apartments on March 2nd? **User:** Sounds good.
ZS-TOD: APICall(method='ScheduleVisit', parameters={ property_name: Golf Club Manor Apartments, visit_date: 2019-03-02 }) ✓
Search Results: [address: 375 Industrial Parkway # 314, has_garage: False, in_unit_laundry: True, number_of_baths: 3, number_of_beds: 3, phone_number: 510-581-0911, price: 3950, property_name: Golf Club Manor Apartments, visit_date: 2019-03-02]
ZS-TOD: Your visit has been scheduled. **User:** I would like to reserve a car there till the 3rd. **ZS-TOD:** Do you like Hatchback, SUV, or another type of car? Would you like to pick up the car from March 2nd and from where? **User:** Sounds good. Delhi Airport is a good place for me to pick it up. A hatchback sounds perfect. **ZS-TOD:** Would you like insurance? What time will you pick it up? **User:** I'd like insurance and will pick it up at 3 PM. **ZS-TOD:** So you want to pick up the car from Indira Gandhi International Airport on March 2nd at 3 PM and drop it off on March 3rd? It's a Hatchback that comes with insurance.
ZS-TOD: APICall: (Missed API Call Here) ✗
Search Results: [Missed Search Results] ✗
User: What is the name of the car and how much is it per day? **ZS-TOD:** It is $39.00 per day and it's called a Fiat Panda. **User:** Thanks for helping. That's it for now. **ZS-TOD:** Have a great day.

C User Study Instructions

Disclaimers of any risks to participants or annotators

There are no significant risks associated with participating in this study. However, annotators may experience mild fatigue or cognitive strain due to prolonged reading and evaluation of multiple conversations. If you feel discomfort or fatigue, please take breaks as needed.

Instructions for Human Study Participants

Your task is to evaluate model-generated responses in multi-turn, task-oriented conversations based on the following criteria:

1. Fluency
2. Informativeness
3. Task Completion

Task Overview

- You will be presented with multiple conversations, where a user interacts with a model to complete a task (e.g., booking a flight).
- Your job is to rate each model response independently using a 1-5 scale based on the provided criteria.
- This study is blind — you will not know which model produced which response.

How to Rate Responses

You will assign a score for each response based on the following:

1. Fluency

Measures whether the response is grammatically correct, well-structured, and natural to read.

- 1 = Unnatural, grammatically incorrect, or hard to read.
- 5 = Perfectly fluent, natural, and error-free.

Example (Good Fluency)

User: I need to book a flight to New York next Monday.
Model: Sure! What time would you like to depart?

Example (Poor Fluency)

User: I need to book a flight to New York next Monday.
Model: Sure! You next Monday flight want time what?

2. Informativeness

Measures whether the response provides useful and relevant information to advance the task.

- 1 = Vague, incorrect, or unhelpful.
- 5 = Precise, relevant, and useful.

Example (Good Informativeness)

User: Can you recommend a vegetarian restaurant nearby?
Model: Yes! Green Leaf Café is a highly rated vegetarian restaurant.

Example (Low Informativeness)

User: Can you recommend a vegetarian restaurant nearby?
Model: There are many restaurants in your area.

3. Task Completion

Measures whether the response successfully progresses or completes the task in the conversation.

- 1 = Fails to address the request.
- 5 = Fully completes the task.

Example (Good Task Completion)

User: I'd like to book a table for two at an Italian restaurant at 7 PM today.
Model: I found a reservation at Bella Italia for 7 PM. Would you like me to reserve it for 2 people at 7 PM today?

Example (Failed Task Completion)

User: I'd like to book a table for two at an Italian restaurant at 7 PM.
Model: There are Italian restaurants in your area.

Additional Guidelines

- Be objective — rate based on quality, not personal preference.
- If a response is unclear or ambiguous, leave a comment.
- Do not try to guess which model produced the response.

Study Duration & Submission

- The study will take approximately 10 minutes to complete.
- Once you have evaluated all responses, submit your ratings.