Evaluating and Enhancing Out-of-Domain Generalization of Task-Oriented Dialog Systems for Task Completion without Turn-level Dialog Annotations

Anonymous ACL submission

Abstract

Traditional task-oriented dialog (ToD) systems rely heavily on labor-intensive turn-level annotations, such as dialogue states and policy labels, for training. This work explores whether large language models (LLMs) can be fine-tuned solely on natural language dialogs to perform ToD tasks, without requiring such annotations. We evaluate their ability to generalize to unseen domains and compare their performance with models trained on fully annotated data. Through extensive experiments with three open-source LLMs of varying sizes and two diverse ToD datasets, we find that models fine-tuned without turn-level annotations generate coherent and contextually appropriate responses. However, their task completion performance—measured by accurate execution of API Calls—remains suboptimal, with the best models achieving only around 42% success in unseen domains. To improve task completion, we propose ZeroToD, a framework that incorporates a schema augmentation mechanism to enhance API Call accuracy and overall task completion rates, particularly in out-of-domain settings. Through neural activation analysis, we show that augmentation enables models to recognize semantic similarities across domains in lower layers while maintaining domain-specific distinctions in higher layers. We also compare ZeroToD with fine-tuning-free alternatives, such as prompting off-the-shelf LLMs, and find that our framework enables smaller, fine-tuned models to outperform large-scale proprietary LLMs in task completion. Additionally, a human study evaluating informativeness, fluency, and task completion confirms our empirical findings. These findings suggest the feasibility of developing cost-effective, scalable, and zero-shot generalizable ToD systems for real-world applications.

1 Introduction

Task-oriented dialog (ToD) systems (Zhang et al., 2020a) enable users to accomplish diverse tasks through natural language interactions. These systems power virtual assistants, customer service chatbots, and various other applications such as making reservations or scheduling appointments (Williams et al., 2016; Zhang et al., 2019b). To be effective, ToD systems must not only engage in user interactions to collect and provide task-specific information but also interface with external systems to accurately complete user tasks.

Traditionally, the development of ToD systems has relied heavily on turn-level manually annotated conversational data, where natural language turns are labeled with dialog states and policy actions (Zhang et al., 2020b). However, this *reliance on turn-level annotated data limits the scalability of ToD systems*, as it prevents them from fully leveraging the vast amounts of readily available unannotated task-oriented conversational data. Furthermore, the annotation process is labor-intensive, expensive, and prone to inconsistencies and errors (Eric et al., 2020; Zang et al., 2020; Han et al., 2021; Budzianowski and Vulic, 2019).

Recent advancements in natural language processing, particularly the emergence of pre-trained large language models (LLMs) (Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019), offer new opportunities to address these scalability challenges. LLMs have demonstrated remarkable capabilities in diverse language tasks, from understanding context to generating coherent responses. While pre-trained models (e.g., GPT-2) have been employed to develop ToD systems (Hosseini-Asl et al., 2020; Yang et al., 2020; Mosharrof et al., 2023b; Budzianowski et al., 2018), their potential to train ToD systems without turn-level annotations remains largely unexplored, as does their *ability to generalize effectively to unseen domains*.

Beyond natural language interactions – such as requesting task-specific information or providing updates – ToD systems must also interact with external systems (e.g., databases) to ensure successful task completion. This often requires retrieving information or executing actions, such as making a reservation via *an API Call*. While ToD systems described in the literature could, in theory, be trained to make such API Calls, this capability *is rarely evaluated in practice* and lacks standardized metrics for proper assessment. The lack of rigorous evaluation and appropriate metrics in this area leaves a significant gap in understanding the readiness of current ToD systems for real-world deployments.

Motivated by the need to evaluate and enhance the out-of-domain generalization of ToD systems, this work investigates three research questions:
RQ1: Can pre-trained LLMs be adapted into effective ToD systems without turn-level annotated data (e.g., annotated dialog states)?
RQ2: Can we improve the out-of-domain generalization of ToD systems for task completion?
RQ3: How does the out-of-domain generalization of fine-tuned ToD systems compare to that of large-scale, proprietary LLMs?

To address RQ1, we frame ToD as a multi-task instruction fine-tuning problem, where the model learns to generate both natural language responses and API Calls by conditioning on the dialog history and domain schema. To enhance task completion performance, we introduce a schema augmentation mechanism that enriches training data with diverse schema variations, significantly improving robustness in unseen domains (RQ2). Finally, to investigate RQ3, we compare fine-tuned ToD systems against fine-tuning-free approaches that rely on large-scale, proprietary LLMs, which are often costly and less controllable.

We conduct extensive experiments on two benchmark ToD datasets – SGD (Rastogi et al., 2019) and KETOD (Chen et al., 2022) – using three open-source models: GPT-2(Radford et al., 2019), Llama-3.2, and FLAN-T5 (Chung et al., 2022). To provide a comprehensive evaluation, we introduce multiple metrics to assess API Call generation, including method name accuracy, parameter correctness, and complete API Call accuracy. For response generation, we use BERTScore (Zhang et al., 2019a) to better capture the semantic similarity between system outputs and ground truth responses. Additionally, we conduct human studies and qualitative analyses on a subset of both datasets to complement automatic evaluations. To provide insights into how schema augmentation enables better out-of-domain performance, we conduct an analysis of neural activation patterns across model layers. Our analysis reveals that augmented models exhibit distinct representational patterns across layers—with early layers capturing shared semantic understanding across domains, while later layers develop domain-specific specialization. This layerwise progression from shared to specialized representations aligns with the model's improved performance in out-of-domain generalization.

Our empirical results provide clear answers to the research questions posed in this study. For RQ1, we compare our approach against state-of-the-art (SOTA) methods that rely on annotated data and find that ToD systems can function effectively without manual annotations by leveraging multi-task instruction fine-tuning. On the complete API accuracy metric, our best model improves by an average of 63.52% across both datasets compared to the strongest baseline SOTA model trained with turn-level annotated data. For RQ2, we evaluate the impact of schema augmentation by comparing models trained with and without this mechanism. Our results show that augmentation significantly enhances out-of-domain generalization, improving complete API accuracy on unseen domains by 17.05% for FLAN-T5 and 35.6% for Llama-3.2 compared to their non-augmented counterparts.

For RQ3, we compare ZeroToD against fine-tuning-free alternatives in unseen domains and confirm that fine-tuning is advantageous for learning when to make API Calls and maintaining strong out-of-domain performance in complex, multi-turn task completion scenarios. On complete API accuracy for unseen domains, FLAN-T5 achieves an average improvement of 43.91% over the best SOTA approach built with the large-scale GPT-4o model. Furthermore, human study results evaluating informativeness, fluency, and task completion closely align with automatic metrics, confirming our empirical findings.

2 Related Work

Pipeline Approaches. ToD systems have traditionally been designed as pipeline systems, where separate components for Natural Language Understanding (NLU), Dialog State Tracking (DST), Dialog Policy, and Natural Language Generation (NLG) are used to handle specific parts of the dialog processing (Ren et al., 2018; Lee, 2013; Peng et al., 2018; Le et al., 2021; Wen et al., 2015; Peng et al., 2020; Chen et al., 2019; Budzianowski et al., 2018; Mosharrof et al., 2023a). However, this approach has drawbacks like error propagation, where errors made in early stages adversely effect modules later on in the pipeline.

End-to-End Approaches. Recent works have shifted towards E2E learning methods, where the ToD task is formulated as conditional generation, where the model generates responses based on the entire dialog history and other relevant annotations (e.g., DST) (Hosseini-Asl et al., 2020; Lin et al., 2021; Bang et al., 2023; Zhang et al., 2023; hyun Ham et al., 2020; Chung et al., 2023a; Yang et al., 2020; Sun et al., 2022a; Imrattanatrai and Fukuda, 2023; Sun et al., 2022b; Zhao et al., 2022; Peng et al., 2021; Mosharrof et al., 2023b; Siddique et al., 2022). A major drawback of these approaches is the dependency on manually annotated data, thus limiting the usage of the wealth of available data. Additionally, most of these approaches assume that API Call results are included in the annotated data, thus limiting their ability to evaluate task completion.

Prompting Approaches. Another recent research direction in ToD systems is in-context learning, where pre-trained LLMs are adapted to specific domains based on contextual examples without requiring fine-tuning (Labruna et al., 2023; Hudevcek and Dusek, 2023; Dingliwal et al., 2021; Madotto and Liu, 2020; Li et al., 2022; Madotto et al., 2021; Chung et al., 2023b; Xu et al., 2024). Even though these approaches show promise on generic domains, they fail on complex domains, and have specialized structure or requirements. Furthermore, the evaluation of API calls in these systems lacks standardized metrics, making it difficult to accurately assess their real-world performance and reliability.

3 Methodology

3.1 Problem Formulation

We formulate ToD task completion as a conditional sequence generation problem, where the system generates natural language responses or API Calls using the dialog history and related domain schemas. We leverage domain schema to facilitate out-of-domain generalization in ToD systems.

We formalize the schema for a given domain $d_x \in D$ by specifying a set of user intents $\mathcal{I}_{d_x}$. For example, in the `Restaurants` domain, one such intent might be `ReserveRestaurant`. Each intent $i_d \in \mathcal{I}_{d_x}$ is then associated with a set of slots $\mathcal{S}_{i_d}$. For instance, `party size` and `reservation time` might be slots for the `ReserveRestaurant` intent. Each slot $s_i \in \mathcal{S}_{i_d}$ is characterized by a tuple $\big(\text{name}(s), \text{is\_required}(s)\big)$, indicating the name of slot (e.g., `reservation date`) and whether it is mandatory to fulfill a desired intent. We denote the entire domain schema for domain d_x as: $\Gamma_{d_x} = \Big(d_x,\, \mathcal{I}_d,\, \{\mathcal{S}_{i_d} \mid i_d \in \mathcal{I}_{d_x}\}\Big)$.

A dialog session $\mathcal{T}_i$ of up to T turns is defined as a sequence of user and system utterances: $\mathcal{T}_i = \big((u_1, r_1), (u_2, r_2), \ldots, (u_T, r_T)\big)$, where u_t is the user utterance and r_t is the system response at turn t. We denote the dialog history at turn t by $H_t = \{(u_1, r_1), (u_2, r_2), \ldots, (u_{t-1}, r_{t-1}), u_t\}$, which encapsulates all user-system exchanges up to and including the current user utterance u_t. Since a single dialog may reference multiple domains, if $\mathcal{T}_i$ spans m domains, we write $\mathcal{T}_i \sim \{d_1, d_2, \ldots, d_m\} \subseteq D$.

3.2 Schema Augmentation

Beyond the original set of domain schemas, we create semantic variations of each domain's intents and slots. Specifically, for each domain $d_x \in D$, we we define its k-th schema variant as: $\Gamma_{\tilde{d}_x^k} = \Big(\tilde{d}_x^k, \tilde{\mathcal{I}}_{d_x^k}, \{\tilde{\mathcal{S}}_{i_d}^k \mid \tilde{i}_d^k \in \tilde{I}_{d_x^k}\}\Big)$, where $\tilde{I}_{d_x^k}$ is the renamed set of intents, and $\tilde{S}_{i_d}^k$ represents the renamed slots for each intent $\tilde{i}_d^k$.

For example, in the `Restaurants` domain, the original intent `ReserveRestaurant` might be changed to `ReserveTable`, and the slot `party size` might become `number of people`. To integrate these augmented schemas into the dialogs, we systematically replace schema references in existing dialogs with their counterparts from $\Gamma_{\tilde{d}_x^k}$. Concretely, for each dialog $\mathcal{T}_i$ associated with domain d_x, we construct an augmented dialog $\tilde{\mathcal{T}}_i^k$ by substituting all intents and slots with those from $\Gamma_{\tilde{d}_x^k}$. This procedure preserves the underlying dialog flow but exposes ZeroToD to multiple schema variations, ultimately improving its ability to generalize to out-of-domain task scenarios.

3.3 Multi-task Instruction Fine-tuning

A ToD system must handle diverse interactions, including general conversation, requesting task-specific information, providing details, and making API Calls for task completion. Broadly, the system generates two types of outputs: (*i*) natural language responses, and (*ii*) API Calls, which include a method name, parameters, and corresponding values. We employ multi-task instruction fine-tuning that trains the model to autonomously decide between generating an API Call or a user response, without introducing special tokens.

Formally, an autoregressive language model (e.g., GPT-2 (Radford et al., 2019)) generates text by predicting the next token given the preceding context. For a given sequence of tokens $(x_1, x_2, \ldots, x_{t-1})$, the probability distribution for the next token x_t is computed as: $p(x_t \mid x_{1:t-1}; \theta) = f_\theta(x_{1:t-1})$, where f_θ represents the model parameterized by θ and outputs a probability distribution over the vocabulary $\mathcal{V}$. The next token x_t is then sampled from this distribution. This formulation extends naturally to response generation in ToD systems, where the system response r_t at turn t is generated recursively until an end-of-sequence token (<eos>) is produced: $r_t \sim p(r_t \mid H_t; \theta)$, where H_t denotes the dialog history up to turn t.

To improve out-of-domain generalization, ZeroToD introduces an additional conditioning variable, the domain schema Γ_{d_x} for each domain d_x and an instruction prompt P. The instructions encourage the model to comprehend schema representations to better generalize across unseen domains and dialog contexts. Extending the above formulation to multi-task instruction fine-tuning for multi-turn dialogs of length T, where each dialog may span multiple domains $\{d_1, d_2, \ldots, d_m\} \subseteq D$, we optimize the following objective: $-\sum_{t=1}^{T} \log p(r_t \mid P, \{\Gamma_{d_j}\}_{j=1}^{m}, H_t, ; \theta)$. Since LLMs operate under a finite context length, the dialog history H_t consists of only the most recent k turns, where $k \leq t$.

3.4 Training Details

The dialog history and domain schema are passed through a structured template to form the inputs to the model. The template is detailed in Figure 3 in Appendix H. Training begins with 500 warm-up steps and early stopping on the evaluation loss with a patience value of 3. We used the AdamW (Loshchilov and Hutter, 2017) optimizer with weight decay and a learning rate of 0.001. Experiments were conducted with GPT2-Medium, FLAN-T5 Large and Llama 3.2 3B Instruct models. GPT-2 and FLAN-T5 were fine-tuned fully, while Llama-3.2 used Low-Rank Adaptation (LoRA) (Hu et al., 2021) and 8-bit quantization (Jacob et al., 2018) for memory efficiency.

Datasets	**SGD**	**KETOD**
# Dialogs	16142	5324
Average Turns / Dialog	20.44	9.78
# Unique API methods:all	46	46
# Unique API methods:unseen	8	8
# Unique API parameters: all	137	134
# Unique API parameters: unseen	88	88

Table 1: Dataset statistics.

4 Experimental Setup

4.1 Datasets

We use two ToD datasets: Schema-Guided Dialog (SGD) dataset, and Knowledge-Enhanced Task-Oriented Dialog (KETOD) dataset. Table 1 shows detailed statistics about the datasets. These datasets are publicly available, large, and represent a wide range of domains that span different tasks. We have selected these datasets as they describe the domain using schema and have the necessary information to simulate communication with external resources through API Calls.

4.2 Evaluation

We evaluate the system across four domain categories: *All Domains* (dialogs from all domains), *Seen Domains* (dialogs from training domains), *Unseen Domains* (dialogs from domains not included in the training data), and *Mixed Domains* (dialogs with both seen and unseen domains).

Response Generation. To evaluate the quality of the response generation of models, we report BERTScore. We used `microsoft/mpnet-base` as the model type for calculating the BERTScore. We report BLEU-4 (Papineni et al., 2002) scores along with additional response generation subtasks—inform and request—in Appendix E.

API Calls. The format for an API Call is: `APICall(method=method_name, parameters =` $\{(s_i, v_i)_{i=1}^{n}\}$). The parameters attribute is a list of slot name and slot value pairs, where s_i represents the slot name and v_i represents the value of that slot.

We use regular expressions to extract different parts of the API Call, and apply custom metrics to assess different parts of an API Call.

Invoke Accuracy measures whether the system can understand when to make an API Call. *Method Accuracy* checks whether the appropriate method name was used in the API Call. *Param Name Accuracy* assesses whether all the parameter names used to construct the API Call are accurately. *Param Value Accuracy* evaluates whether each parameter value corresponding to a parameter name is correct. It is important to note that this metric will

Dataset	Model	Annotations	Overall Response (BertScore-F1)				Complete API Accuracy			
			all	seen	mixed	unseen	all	seen	mixed	unseen
SGD	SOLOIST	Yes	0.6214	0.6538	0.6265	0.6097	19.25	47.82	24.32	07.66
	InstructTODS	Yes	0.5960	0.5926	0.6014	0.5912	09.25	09.50	09.85	08.57
	SimpleTOD	Yes	0.6100	0.600	0.6300	0.5800	20.30	44.94	25.57	09.39
	Q-TOD	Yes	0.2351	0.2308	0.2431	0.2287	21.73	44.78	27.39	10.76
	ZS-ToD	Yes	0.5704	0.6439	0.5648	0.5600	20.38	56.15	20.28	12.62
	SyncTOD	Yes	0.4872	0.4969	0.4888	0.4838	24.64	25.23	29.95	19.12
	GPT-2	No	0.7002	0.7291	0.7149	0.6800	35.53	74.77	42.87	19.24
	Llama-3.2	No	0.7629	**0.7850**	0.7708	0.7507	52.84	**90.19**	57.44	39.84
	FLAN-T5	No	**0.7633**	0.7792	**0.7723**	**0.7513**	**65.87**	89.88	**72.99**	**53.16**
KETOD	SOLOIST	Yes	0.5035	0.5201	0.4895	0.4983	24.12	43.79	17.29	05.98
	InstructTODS	Yes	0.4826	0.4834	0.4904	0.4731	08.86	09.06	08.38	09.20
	SimpleTOD	Yes	0.5248	0.5540	0.5382	0.4735	36.24	61.07	31.19	08.74
	Q-TOD	Yes	0.6657	0.6800	0.6625	0.6513	19.35	31.54	15.15	08.05
	ZS-ToD	Yes	0.4759	0.4822	0.4643	0.4809	26.70	43.29	21.75	10.34
	SyncTOD	Yes	0.4930	0.4956	0.4973	0.4852	13.44	13.93	15.69	09.89
	GPT-2	No	0.6766	0.7001	0.6821	0.6410	36.75	59.56	32.62	10.80
	Llama-3.2	No	0.7369	0.7624	0.7363	0.7057	63.32	91.61	55.08	35.17
	FLAN-T5	No	**0.7431**	**0.7665**	**0.7457**	**0.7112**	**77.26**	**95.97**	**75.58**	**53.79**

Table 2: Performance comparison between annotation-dependent baselines and annotation-free ZeroToD models on Overall Response and API Accuracy (RQ1).

only be considered if the corresponding parameter name is correct. *Complete API Call Accuracy* metric checks whether the complete API Call (i.e., all components) was generated correctly.

4.3 Baselines

AutoTOD (Xu et al., 2024) introduced a zero shot autonomous ToD agent, that works without manual annotations and also has the ability to communicate with external resources.

SyncTOD (Saley et al., 2024) proposed a ToD model that employs small auxiliary models to provide task-specific hints and select exemplars for in-context prompts.

InstructTODS (Chung et al., 2023b) presented a zero-shot E2E ToD model that generates proxy belief state that guides the model for response generation.

ZS-TOD (Mosharrof et al., 2023b) introduced a zero-shot generalizable E2E ToD model that incorporates domain schema and dialog annotations to generate dialog responses.

Q-TOD (Tian et al., 2022) introduced a query-driven TOD system that extracts essential information from the dialog context into a query and uses it to retrieve relevant knowledge records for response generation.

SimpleTOD (Chen et al., 2022) introduced a ToD model as an end-to-end sequence generation problem that utilizes the dialog history, dialog states and system actions to generate system responses.

SOLOIST (Peng et al., 2021) introduced an E2E ToD system that employs a transformer-based autoregressive model that generates dialog responses grounded in user goals and real-world knowledge for task completion.

SOLOIST, SimpleTOD and ZS-ToD were implemented using GPT-2 Medium. During inference, we extract the system response and disregard the additional information like dialog state and system actions.

5 Results

Table 2 presents the findings for *RQ1: Can pre-trained LLMs be adapted into effective ToD systems without turn-level annotated data ?* Turn-level annotations include detailed labeling of dialog states, system actions, and responses for each conversation turn, requiring significant manual effort. In contrast, API annotations only indicate the presence and structure of API calls in the dialog, which are inherently available in task-oriented conversations. This distinction is crucial as ZeroToD models leverage only API annotations while avoiding the need for extensive turn-level labeling.

Our results show that ZeroToD models, which do not rely on turn-level annotations, outperform models trained with annotated data in response generation. A key reason for this improvement is that ZeroToD models focus solely on generating system responses, whereas annotation-based models must produce structured outputs that include dialog state, system actions, and responses—requiring the model to optimize for multiple complex tasks simultaneously. Furthermore, the substantial performance gap between the baseline approaches built with GPT-2 and the GPT-2 variant of ZeroToD suggests that learning to generate responses directly is a more effective approach for ToD systems.

Dataset	Model	Augm-ented	Complete API Accuracy			
			all	seen	mixed	unseen
SGD	GPT-2	✗	35.53	74.77	42.87	19.24
	GPT-2	✓	47.66	82.01	53.76	33.75
	Llama-3.2	✗	52.84	90.19	57.44	39.84
	Llama-3.2	✓	62.38	**94.31**	63.99	53.68
	FLAN-T5	✗	65.87	89.88	72.99	53.16
	FLAN-T5	✓	**73.49**	90.65	**81.76**	**61.07**
KETOD	GPT-2	✗	36.75	59.56	32.62	10.80
	GPT-2	✓	48.43	72.48	40.64	25.52
	Llama-3.2	✗	63.32	91.61	55.08	35.17
	Llama-3.2	✓	73.24	**97.48**	67.02	48.05
	FLAN-T5	✗	77.26	95.97	75.58	53.79
	FLAN-T5	✓	**82.66**	96.48	**82.35**	**64.14**

Table 3: Impact of Schema Augmentation Mechanism on API Accuracy (RQ2).

Dataset	Model	Overall Response	API Invoke Accuracy	Complete API Accuracy
SGD	Auto-ToD	0.5471	63.15	42.20
	GPT-2	0.7068	91.26	33.75
	Llama-3.2	**0.7506**	**98.80**	53.68
	FLAN-T5	0.7196	94.42	**61.07**
KETOD	Auto-ToD	0.5471	63.22	41.61
	GPT-2	0.6372	94.48	25.52
	Llama-3.2	**0.6454**	**97.70**	48.05
	FLAN-T5	0.7050	92.18	**59.54**

Table 4: Evaluation of fine-tuned approaches against large-scale proprietary LLMs on unseen domains (RQ3).

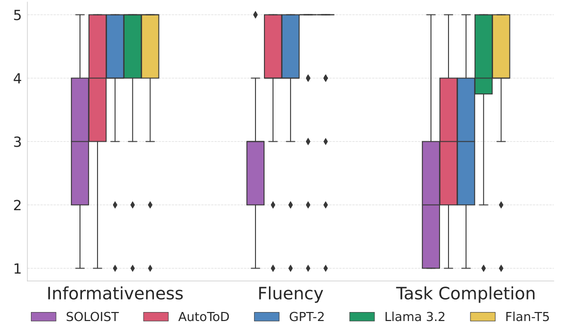


Figure 1: Human Evaluation Study on SGD and KETOD. Evaluators were asked to rate the dialog samples between a range of 1-5 on 3 categories.

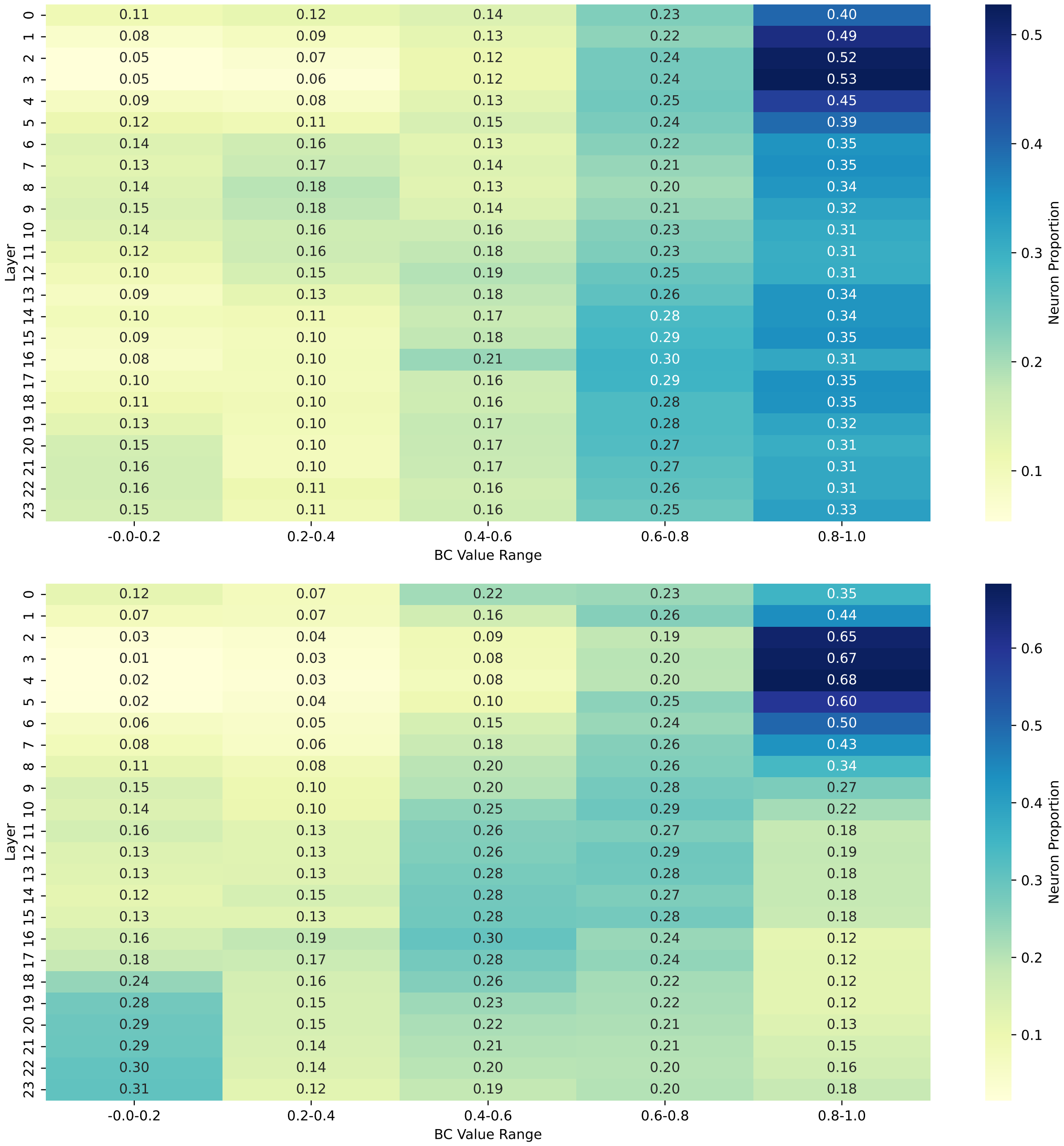
For task completion, all models trained without turn-level annotations consistently outperform the annotated models. This finding highlights the sufficiency of dialogue history as a standalone source of context for completing complex tasks. Table 2 reveals more insights about the different ZeroToD models. FLAN-T5 and Llama-3.2 being the larger models, significantly outperform the smaller GPT-2 model for task completion. However, even though Llama-3.2 is a larger model than FLAN-T5, it does not have better task completion performance.

This discrepancy may stem from differences in the training methodologies. Specifically, Llama-3.2 was trained using 8-bit quantization and LoRA adapters, whereas FLAN-T5 underwent full fine-tuning. The use of LoRA significantly reduces the number of trainable parameters and the 8-bit quantization introduces precision loss due to the reduced bit width. These factors likely contributed to Llama-3.2's lower performance despite its larger model size.

Schema Augmentation Performance. Table 3 presents the results for *RQ2: Can we improve the out-of-domain generalization of ToD systems for task completion?*. Across all the models, we can see that the response generation performance is similar, but there are improvements in task completion performance, specially a big increment in the unseen domain. For seen domains, there is a small improvement, which is expected as the augmentation mainly teaches the models how to use the schema to generalize to out-of-domain data, however for unseen domains, this learning is very useful and the models have shown considerable improvements. Between Llama-3.2 and FLAN-T5, we can see that for seen domains Llama-3.2 has a slightly better performance however for unseen domains Llama-3.2 has much lower performance. One reason for this could be the size of the two models, Llama-3.2 being the larger model may have a higher capacity to memorize the training data, which could explain its stronger performance on seen domains. However, this can also make it more prone to over-fitting and may not generalize well to new, unseen domains.

Neuron Activation Analysis. In ToD systems, it is common for semantically similar concepts to appear across different domains. To understand how augmentation enables better out-of-domain generalization, we analyzed the internal representations learned by FLAN-T5 for semantically similar slots across domains: `location (Restaurants)`, `where_to (Hotels)`, and `destination (Ridesharing)`. While these slots represent the same semantic concept of location, they manifest differently depending on the domain context. We recorded neural activations from all decoder layers during the generation process by implementing forward hooks at each decoder block, capturing hidden states corresponding to these slot name tokens. The activations were visualized using Kernel Density Estimation (KDE) plots to understand the distribution patterns across these semantically related slots. Technical details about the hidden state capture process and methodology are provided in Appendix D.

To quantify the model's ability to develop generalized representations, we employed the Bhattacharyya Coefficient (BC) (Bhattacharyya, 1943),



(b) Augmented model

Figure 2: Distribution of Neuron Proportions Across Bhattacharyya Coefficient(BC) Ranges in Model Layers. The heatmaps show the proportion of neurons (values between 0-1) exhibiting different ranges of BC values (0.0-1.0) across all 24 layers of FLAN-T5. Higher BC values indicate more overlap in representations, while lower values suggest more distinct patterns. The augmented model shows a clear transition from shared representations in early layers to more specialized patterns in later layers.

which measures the overlap between probability distributions. The BC metric ranges from 0 to 1, where values closer to 0 indicate distinct patterns and values closer to 1 suggest overlapping distributions. We binned the BC values into five ranges and visualized their distribution across all layers using frequency distribution heatmaps.

The comparison between non-augmented (Fig 2a) and augmented (Fig 2b) models reveals how schema augmentation influences the model's internal representations. In the lower layers (0-7), the augmented model shows higher BC values (0.60-0.68 in layers 2-5), indicating that the activations are forming similar distributions for these location-related slots, suggesting the model recognizes their shared semantic meaning despite different domain origins.

The middle layers (8-15) of the augmented model show decreasing BC values compared to the non-augmented model, indicating that the distributions begin to diverge as the model develops more domain-specific representations. Most significantly, in the higher layers (16-23), the augmented model shows a shift towards lower BC values (0.28-0.31 in 0.0-0.2 range), indicating it has learned to differentiate and specialize representations based on domain-specific context. In contrast, the non-augmented model (Fig 2a) maintains higher BC values throughout, suggesting less ability to distinguish between domain-specific uses of location information.

This transition from shared representations in early layers to specialized processing in later layers suggests that augmentation helps the model develop a more robust understanding of semantically similar concepts while maintaining the flexibility to adapt to domain-specific requirements.

Fine-tuning Performance. Table 4 presents the results on unseen domains for ZeroToD models, and Auto-ToD, which was built using GPT-4o. Using the results in Table 4, we can answer *RQ3: How does the out-of-domain generalization of fine-tuned ToD systems compare to that of large-scale, proprietary LLMs?* For the Complete API Accuracy metric, except for the GPT-2 model, all other ZeroToD models outperform Auto-ToD. For all the other metrics, Auto-ToD has much lower scores than the ZeroToD models. A key metric to note here is the API Invoke Accuracy, which measures whether a model is making an API call on the right turn, and Auto-ToD has a very low score on this metric when compared to ZeroToD models.

Due to this issue, Auto-ToD also has a much lower score for the Overall Response metric, as it makes API Calls on turns where a general interaction is expected. Specifically, ZeroToD models show substantial improvements over Auto-ToD: on SGD, Llama-3.2 achieves a 37.2% relative improvement in Overall Response (0.7506 vs 0.5471) and a 56.4% improvement in API Invoke Accuracy (98.80 vs 63.15). Based on these results, we can state that fine-tuning is an important step to identify the timing of making an API Call in ToD systems.

Further analysis in Appendix F reveals another interesting pattern: while Auto-ToD performs well on simple domains like `Alarm` and `Movies`, it struggles with complex domains such as `Restaurants` and `Buses`. In contrast, ZeroToD models maintain consistent performance across both simple and complex domains, demonstrating the robustness achieved through fine-tuning.

Human Evaluation. To supplement the automatic metrics and get a qualitative analysis, we conducted a human evaluation using Amazon Mechanical Turk to assess the performance of various models. Two baseline models (SOLOIST and Auto-ToD) and three ZeroToD models (GPT-2, Llama-3.2, and FLAN-T5) were taken into account. We sampled 100 dialogs from each dataset, with 50 coming from single-domain tasks and the remaining 50 from multi-domain tasks, all from the test dataset. Human evaluators were asked to rate the models on a scale from 1 to 5 on three questions: the accuracy of information presented in the responses (Informativeness), how fluent and natural the conversation is (Fluency), and whether the models can make accurate API Call (Task Completion).

The results, shown in Figure 1, align with the automatic metrics, where ZeroToD models outperform the existing SOTA approaches. This demonstrates a strong alignment between quantitative and qualitative assessments. Notably, for task completion and fluency, Llama-3.2 and FLAN-T5 demonstrate superior performance compared to all other models, which is consistent with our previous findings. Another important observation is that Llama-3.2 and FLAN-T5 have less variance in performance across all tasks when compared to all other models, which further solidifies the robustness of our approach.

6 Conclusion

This work demonstrates that LLMs fine-tuned solely on natural language dialogs can effectively generalize to unseen domains by framing ToD as a multi-task instruction fine-tuning problem. To further enhance their out-of-domain task completion performance, we introduce schema augmentation, which improves model adaptability to unseen domains and strengthens task completion performance. To ensure robust evaluation of task completion, we explicitly incorporate API Calls as a core task and assess performance using both automatic metrics and human evaluations. Furthermore, we show that fine-tuned ToD systems generalize better to unseen domains than fine-tuning-free approaches that rely on large-scale proprietary LLMs. These results highlight the feasibility of developing cost-effective, scalable, and zero-shot generalizable ToD systems that achieve strong out-of-domain generalization without requiring turn-level annotations, paving the way for their practical adoption in real-world applications.

7 Limitations

ZeroToD has been developed by fine-tuning LLMs such as GPT-2, Llama-3.2, and FLAN-T5. These LLMs require significant computational resource requirements to train, particularly Llama-3.2. Training and inference with these models can be expensive, limiting their practicality for deployment in resource-constrained environments.

The LLMs used in the system function as black boxes, making it challenging to interpret the reasoning behind their responses. This lack of transparency hinders the ability to diagnose and correct erroneous outputs, which is crucial in ToD systems where accuracy is critical. Furthermore, the models may inherit biases present in the training data, leading to biased or unfair responses in certain scenarios. Although efforts were made to mitigate this issue by fine-tuning using the dialog datasets, completely eliminating biases remains a challenging task. The reliance on pre-trained models introduces limitations related to the coverage of the pre-training data. If the pre-training data lacks specific domain knowledge, the ToD system may under perform in those domains.

The deployment of LLMs in ToD systems raises ethical and privacy concerns, particularly regarding the handling of sensitive user data. Ensuring that the system complies with privacy regulations and ethical standards is an ongoing challenge that requires continuous monitoring and updates. Similar to other AI technologies, there is a scope for potential misuse of our system. If ZeroToD is used with malicious intent or the model is fed inappropriate data, there is a risk of abuse. We would strongly advise to take necessary precautions and appropriate usage policies.

Addressing the limitations outlined above is crucial for advancing the effectiveness and reliability of ToD systems. While the usage of pre-trained LLMs offers significant advantages, these models are not without their challenges. Increasing model interpretability, mitigating biases, and addressing ethical and societal concerns are essential steps toward creating more robust and responsible ToD systems.

References

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-end task-oriented dialogue system. *ArXiv*, abs/2305.02468.

Anil Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distribution. *Bulletin of the Calcutta Mathematical Society*, 35:99–110.

Pawel Budzianowski, Iñigo Casanueva, Bo-Hsiang Tseng, and Milica Gasic. 2018. Towards end-to-end multi-domain dialogue modelling.

Paweł Budzianowski and Ivan Vulic. 2019. Challenges in building intelligent open-domain dialog systems. In *Proceedings of the 2019 International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6429–6433. IEEE.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. *arXiv preprint arXiv:1905.12866*.

Zhiyu Chen, Bing Liu, Seungwhan Moon, Chinnadhurai Sankar, Paul A. Crook, and William Yang Wang. 2022. Ketod: Knowledge-enriched task-oriented dialogue. *ArXiv*, abs/2205.05589.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Willy Chung, Samuel Cahyawijaya, Bryan Wilie, Holy Lovenia, and Pascale Fung. 2023a. Instructtods: Large language models for end-to-end task-oriented dialogue systems. *ArXiv*, abs/2310.08885.

Willy Chung, Samuel Cahyawijaya, Bryan Wilie, Holy Lovenia, and Pascale Fung. 2023b. Instructtods: Large language models for end-to-end task-oriented dialogue systems. *arXiv preprint arXiv:2310.08885*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Saket Dingliwal, Bill Gao, Sanchit Agarwal, Chien-Wei Lin, Tagyoung Chung, and Dilek Z. Hakkani-Tür. 2021. Few shot dialogue state tracking using meta-learning. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Mihail Eric, Rahul Goel, Shachi H Paul, Abhinav Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428.

Ting Han, Chia-Chun Wang, Guan-Lin Chao, Zihan Liu, Ryuichi Takanobu, Hua Xu, and Minlie Huang. 2021. Multiwoz 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation. *arXiv preprint arXiv:2010.05594*.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. Simpletod: A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems (NeurIPS)*.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Vojtvech Hudevcek and Ondrej Dusek. 2023. Are large language models all you need for task-oriented dialogue? In *SIGDIAL Conferences*.

Dong hyun Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kyungmin Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2. In *Annual Meeting of the Association for Computational Linguistics*.

Wiradee Imrattanatrai and Ken Fukuda. 2023. End-to-end task-oriented dialogue systems based on schema. In *Annual Meeting of the Association for Computational Linguistics*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713.

Tiziano Labruna, Sofia Brenna, Andrea Zaninello, and Bernardo Magnini. 2023. Unraveling chatgpt: A critical analysis of ai-generated goal-oriented dialogues and annotations. In *International Conference of the Italian Association for Artificial Intelligence*.

Nhat Le, AB Siddique, Fuad Jamour, Samet Oymak, and Vagelis Hristidis. 2021. Predictable and adaptive goal-oriented dialog policy generation. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 40–47. IEEE.

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

Zekun Li, Wenhu Chen, SHIYANG LI, Hong Wang, Jingu Qian, and Xi Yan. 2022. Controllable dialogue simulation with in-context learning. *ArXiv*, abs/2210.04185.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A. Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. *ArXiv*, abs/2105.04222.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *ArXiv*, abs/2110.08118.

Andrea Madotto and Zihan Liu. 2020. Language models as few-shot learner for task-oriented dialogue systems. *ArXiv*, abs/2008.06239.

Adib Mosharrof, Moghis Fereidouni, and A.B. Siddique. 2023a. Toward open-domain slot filling via self-supervised co-training. *Proceedings of the ACM Web Conference 2023*.

Adib Mosharrof, Muhammad Hasan Maqbool, and A.B. Siddique. 2023b. Zero-shot generalizable end-to-end task-oriented dialog system using context summarization and domain schema. *ArXiv*, abs/2303.16252.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics*.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Lidén, 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, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*.

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. *arXiv preprint arXiv:2002.12328*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *AAAI Conference on Artificial Intelligence*.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. *arXiv preprint arXiv:1810.09587*.

Vishal Vivek Saley, Rocktim Jyoti Das, Dinesh Raghu, et al. 2024. Synergizing in-context learning with hints for end-to-end task-oriented dialog systems. *arXiv preprint arXiv:2405.15585*.

A.B. Siddique, Muhammad Hasan Maqbool, Kshitija Taywade, and Hassan Foroosh. 2022. Personalizing task-oriented dialog systems via zero-shot generalizable reward function. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2022a. Bort: Back and denoising reconstruction for end-to-end task-oriented dialog. In *NAACL-HLT*.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2022b. Mars: Semantic-aware contrastive learning for end-to-end task-oriented dialog. *ArXiv*, abs/2210.08917.

Xin Tian, Yingzhan Lin, Mengfei Song, Siqi Bao, Fan Wang, Huang He, Shuqi Sun, and Hua Wu. 2022. Q-tod: A query-driven task-oriented dialogue system. *arXiv preprint arXiv:2210.07564*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

J. Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue Discourse*, 7:4–33.

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 *Annual Meeting of the Association for Computational Linguistics*.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2020. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. *ArXiv*, abs/2012.03539.

Xiaoxue Zang, Qi Zhu, Jiachen Zhang, Zheng-Yu Li, Baolin Peng, Xiujun Li, Liangjun Zhou, Jinchao Li, Jianfeng Gao, and Yun-Nung Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117.

Jianguo Zhang, Stephen Roller, Kun Qian, Zhiwei Liu, Rui Meng, Shelby Heinecke, Haiquan Wang, Silvio Savarese, and Caiming Xiong. 2023. Enhancing performance on seen and unseen dialogue scenarios using retrieval-augmented end-to-end task-oriented system. In *SIGDIAL Conferences*.

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

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2019b. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *AAAI Conference on Artificial Intelligence*.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020a. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

Zheng Zhang, Ryuichi Takanobu, Minlie Huang, and Xiaoyan Zhu. 2020b. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63:2011 – 2027.

Jeffrey Zhao, Yuan Cao, Raghav Gupta, Harrison Lee, Abhinav Rastogi, Mingqiu Wang, Hagen Soltau, Izhak Shafran, and Yonghui Wu. 2022. Anytod: A programmable task-oriented dialog system. In *Conference on Empirical Methods in Natural Language Processing*.

Dataset	Model	Augm-ented	Overall Response (BertScore-F1) all	seen	mixed	unseen
SGD	GPT-2	✗	0.7002	0.7291	0.7149	0.6800
	GPT-2	✓	0.7266	0.7431	0.7437	0.7068
	Llama-3.2	✗	0.7629	0.7850	0.7708	0.7357
	Llama-3.2	✓	0.7623	**0.7852**	0.7693	0.7506
	FLAN-T5	✗	**0.7633**	0.7792	**0.7723**	**0.7513**
	FLAN-T5	✓	0.7320	0.7494	0.7411	0.7396
KETOD	GPT-2	✗	0.6766	0.7001	0.6821	0.6410
	GPT-2	✓	0.6677	0.6867	0.6738	0.6372
	Llama-3.2	✗	0.7369	0.7624	0.7363	0.7057
	Llama-3.2	✓	0.7405	0.7679	0.7389	0.7082
	FLAN-T5	✗	0.7431	0.7665	0.7457	0.7112
	FLAN-T5	✓	**0.7549**	**0.7786**	**0.7541**	**0.7261**

Table 5: Impact of Schema Augmentation Mechanism on Response Generation (RQ2).

A Template for multi-task instruction fine-tuning

Figure 3 shows the template used to process the inputs for ZeroToD. The template first informs about the domains involved in the dialog. Next, it provides task-specific instructions and provides the domain schemas. The dialog history is provided and the model is instructed to generate a system response based on the dialog history, search results, and the task instructions.

> You are an expert chat assistant for the domains: [domains].
>
> Instructions: As an expert, you must generate the most appropriate response for the chat assistant. The response can be an api call or a response to the user.
>
> Based on the Last User Utterance, you must find the relevant Intent from the Schema and your request should use the required slots and optional slots from that Intent.
>
> You will be provided with the Schema for domains: [domains]
>
> [schemas]
>
> You will be provided an incomplete dialog between a user and a chat assistant, and an optional search results.
>
> Dialog History: [dialog history]
>
> Using the Dialog History, Search Results, and by following the Instructions please generate the response for the chat assistant.

Figure 3: Multi-task instruction finetuning template. Items in blue are dynamic elements and those in purple are important aspects of the prompt.

B Impact of Schema Augmentation on Response Generation (RQ2).

Table 5 showcases the effect of schema augmentation on response generation performance for ZeroToD models across the SGD and KETOD dataset. With augmentation, models are exposed to different schemas and thus encounter a more diverse set of data. This exposure is primarily intended to improve Task Completion and API Call Accuracy, rather than directly enhancing response quality. Therefore, we expect the response generation performance to remain largely unchanged or to show only modest improvements. This pattern is indeed reflected Table 5. Across both datasets, the differences in response generation scores between the augmented and non-augmented models are generally small.

C Detailed Task Completion Performance.

To complete a task, a model has to make a correct API Call. An API Call has many aspects in it, and we present detailed results in Table 6. We can see that ZeroToD models considerably outperform baseline

Dataset	Model	Augm-ented	API Invoke Accuracy				API Method Accuracy				Param Names Accuracy				Param Values Accuracy			
			all	seen	mixed	unseen	all	seen	mixed	unseen	all	seen	mixed	unseen	all	seen	mixed	unseen
SGD	SOLOIST	✗	79.92	80.37	80.29	79.44	64.51	72.82	66.38	60.74	47.06	72.05	52.84	35.54	45.16	71.30	51.18	33.13
	SimpleTOD	✗	69.17	60.51	71.55	68.59	56.89	53.50	59.54	54.88	41.11	51.62	45.38	34.33	39.53	51.06	43.70	32.64
	Q-TOD	✗	49.08	51.71	50.78	46.73	42.33	50.86	44.51	38.18	34.45	49.40	37.44	28.04	33.83	49.14	36.91	27.25
	ZS-ToD	✗	84.98	90.11	84.09	84.78	63.57	79.98	62.17	61.43	44.44	73.15	41.50	40.17	42.74	72.22	40.04	38.13
	GPT-2	✗	86.67	93.15	88.72	83.10	75.98	91.12	79.19	69.30	59.80	86.96	63.20	50.27	57.53	85.89	61.34	47.31
	GPT-2	✓	93.89	96.81	95.79	91.26	83.07	89.17	86.72	77.92	76.94	91.71	78.41	72.15	74.18	90.86	76.02	68.59
	Llama-3.2	✗	98.08	**99.69**	98.28	97.52	92.18	**99.69**	91.73	91.00	84.53	98.36	85.54	80.44	81.55	97.92	82.90	76.55
	Llama-3.2	✓	**98.78**	99.38	98.62	**98.80**	95.26	99.38	94.88	94.97	89.23	**99.10**	88.22	88.11	86.25	**98.73**	85.77	84.02
	FLAN-T5	✗	97.93	99.77	98.53	96.90	**97.61**	99.77	98.53	**96.90**	88.64	97.49	90.82	84.42	86.24	97.20	89.01	80.95
	FLAN-T5	✓	96.28	99.53	97.36	94.42	95.97	99.53	97.20	93.89	**94.37**	97.23	95.87	**92.17**	**92.08**	96.99	94.12	**88.86**
KETOD	SOLOIST	✗	51.57	53.36	48.13	53.56	45.16	51.51	41.35	41.38	36.69	49.87	29.86	27.44	35.47	49.34	28.34	25.64
	SimpleTOD	✗	73.81	74.50	78.97	66.21	65.14	71.98	69.34	50.34	51.83	68.59	51.27	29.57	50.40	67.88	49.32	27.82
	Q-TOD	✗	32.41	34.56	32.44	29.43	29.52	34.06	29.23	23.68	26.59	34.39	23.35	20.08	26.14	34.29	22.59	19.54
	ZS-ToD	✗	72.42	70.47	72.01	75.63	56.97	61.24	52.41	57.01	44.58	56.51	38.74	35.78	43.30	55.84	37.45	33.65
	GPT-2	✗	78.83	80.87	80.75	73.56	71.48	78.36	71.12	62.53	57.02	72.98	53.57	39.60	55.46	71.83	51.83	37.73
	GPT-2	✓	92.96	91.78	93.05	94.48	86.62	88.09	86.10	85.29	75.66	84.53	70.67	69.97	72.85	83.55	67.55	65.02
	Llama-3.2	✗	96.55	97.48	96.61	95.17	92.90	96.98	91.62	88.97	86.04	97.13	81.70	76.43	84.26	96.65	79.54	73.37
	Llama-3.2	✓	**98.49**	**99.16**	98.40	**97.70**	**96.80**	**99.16**	95.75	**94.94**	**91.86**	**98.67**	89.12	86.05	**90.10**	**98.62**	86.69	82.83
	FLAN-T5	✗	98.62	99.33	98.75	97.47	98.49	99.33	98.75	97.01	92.89	98.67	92.08	86.02	91.39	98.38	90.18	83.40
	FLAN-T5	✓	97.80	98.99	97.86	96.09	97.61	98.99	97.68	95.63	95.91	98.44	95.58	**92.88**	94.35	98.24	93.65	**89.92**

Table 6: Additional API Metrics for baseline approaches and ZeroToD models (RQ1, RQ2).

approaches across all metrics. Upon inspecting the API Call Invoke Accuracy, we see that baseline approaches have much lower scores, indicating that they struggle in identifying when to make API Calls. The API Call Method Accuracy evaluates whether a model generates the correct method name in the API Call. A common pattern that we see across all models is that there is a drop in parameter names accuracy when compared to the previous metrics. Generating the correct list of parameters for the API Call is inherently a harder problem than deciding when to make an API Call and what method to use, so the performance degradation is understandable.

A key observation from Table 6 is the significant impact of the schema augmentation on the API Call parameter names metric. Our results indicate that schema augmentation yields the largest improvement for this metric. API Call parameters are directly derived from the schema, and schema augmentation enables the models to better recognize and utilize these patterns, thus improving the model's ability to generate the correct list of parameters, leading to a notable increase in parameter names accuracy. Furthermore, the API Call parameter values accuracy also improved as a result, since a model is only rewarded for generating the correct value if it is assigned to the appropriate parameter name.

For instance, consider the task of finding a bus using the `FindBus` method. We compare two schema variations, `Buses_1` and `Buses_11`, which define different slot names for the same concepts. In `Buses_1`, the slot names are `from_station` and `to_station`, and for `Buses_11`, the slot names are `origin` and `destination`.

A model trained without schema augmentation tends to overfit to specific slot names seen during training. If the model was trained on `Buses_1`, it might always generate `from_station` and `to_station`, even when interacting with `Buses_11`, leading to incorrect API Calls. For example, given the user utterance: *"I want to find a bus from LA to SFO"*, the model without augmentation might generate:

```
API Call(method=FindBus, parameters={from_station=LA, to_station=SFO }).
```

In the `Buses_11` schema, the slot names `from_station` and `to_station` do not exist, thus making the API Call invalid.

On the other hand, a model trained with schema augmentation learns to generalize across schema variations by recognizing slot name patterns from multiple schemas, and might generate:

```
API Call(method=FindBus, parameters={origin=LA, destination=SFO}).
```

The model can dynamically align its output with the schema it is conditioned on. By learning to use the slot names from the provided schema rather than relying on the memorized slot names, a model trained with schema augmentation demonstrates improved robustness and generalization.

D Technical details of Model Activation Analysis

Technical Details of Neural Activation Capture. We implemented forward hooks at each decoder block to capture neural activations during the generation process. The activation capture was performed using PyTorch's register_forward_hook functionality:

```python
def __init__(self, model, tokenizer):
    super().__init__(model, tokenizer)
    n_layers = len(self.model.decoder.block)
    self.layer_names = [f"block.{i}" for i in range(n_layers)]

    # Register hooks for each layer
    for layer_idx, layer in enumerate(self.model.decoder.block):
        layer.register_forward_hook(self._capture_activations(layer_idx))
```

For each layer in the T5 decoder (24 layers total), we recorded hidden states with dimension size 1024. The activation capture function was implemented as:

```python
def _capture_activations(self, layer_idx):
    def hook(module, input, output):
        if isinstance(output, tuple):
            hidden_states = output[0]
        else:
            hidden_states = output

        if layer_idx not in self.activations:
            self.activations[layer_idx] = []
        self.activations[layer_idx].append(hidden_states)
        return output
    return hook
```

The captured activations were stored in a tensor of shape (batch_size, sequence_length, n_layers, hidden_dim), where batch_size represents the number of examples, sequence_length is the length of the generated sequence, n_layers is the number of decoder layers (24), and hidden_dim is the model's hidden dimension (1024). These recorded activations were then used to compute the Bhattacharyya Coefficient (BC) values across different layers to analyze the model's internal representations of semantically similar slots across domains.

E BLEU Scores for Response Generation

Table 7 presents BLEU-4 scores for response generation and its subtasks inform and request. We have used augmented ZeroToD models.

The Overall Response metric is computed for all turns in which the system is expected to generate a response, providing a broad indicator of the model's fluency and relevance. For our analysis, we break down response generation into two distinct subtasks: inform and request, which help us better evaluate different aspects of the system's response capabilities.

The Inform subtask focuses on dialogue turns where the system must provide specific information requested by the user. For example, when the user asks for the address of a restaurant, the system must accurately generate that address in its response. This subtask is typically easier because the data is already available to the system; the main challenge lies in integrating the correct information into a well-formed and contextually appropriate response.

In contrast, the Request subtask involves dialogue turns where the system must ask the user for additional information to complete the task. For instance, in the restaurant booking scenario, if the user does not specify the reservation time, the system must generate a clarifying question such as, "What time would you like to make the reservation?" This subtask is more challenging than Inform because there can be multiple reasonable ways to formulate a request in a given context, yet we compare the generated request against the specific ground truth in the dataset.

We see a similar trend here as well, with ZeroToD models outperforming baseline approaches. Overall, our BLEU results highlight that the Request subtask tends to be more difficult than Inform, as reflected

Dataset	Model	Overall Response (BLEU-4)				Inform (BLEU-4)				Request (BLEU-4)			
		All	Seen	Mixed	Unseen	All	Seen	Mixed	Unseen	All	Seen	Mixed	Unseen
SGD	SimpleTOD	0.1696	0.1834	0.1877	0.1494	0.1685	0.1790	0.1896	0.1438	0.0228	0.0195	0.0216	0.0243
	SOLOIST	0.1902	0.2798	0.1990	0.1655	0.1813	0.2226	0.1945	0.1568	0.0281	0.0339	0.0265	0.0284
	InstructTODS	0.0727	0.0729	0.0767	0.0690	0.1460	0.1380	0.1506	0.1432	0.0074	0.0051	0.0071	0.0080
	SyncTOD	0.0563	0.0645	0.0584	0.0528	0.1047	0.1072	0.1090	0.0997	0.0111	0.0131	0.0106	0.0111
	Q-TOD	0.0393	0.0369	0.0430	0.0364	0.0946	0.0919	0.1030	0.0858	0.0061	0.0065	0.0059	0.0061
	ZS-ToD	0.0590	0.1413	0.0568	0.0512	0.0255	0.0402	0.0228	0.0246	0.0231	0.0367	0.0221	0.0215
	Auto-ToD	0.0487	0.0523	0.0501	0.0466	0.0854	0.0743	0.0884	0.0851	0.0173	0.0111	0.0159	0.0195
	GPT-2	0.2015	0.2109	0.2229	0.1802	0.2181	0.2421	0.2368	0.1923	0.0400	0.0275	0.0423	0.0403
	Llama-3.2	**0.2445**	**0.2905**	**0.2568**	**0.2242**	**0.2888**	**0.3180**	**0.3043**	**0.2650**	**0.0641**	**0.0803**	**0.0614**	**0.0634**
	FLAN-T5	0.2110	0.2332	0.2226	0.1961	0.2811	0.3098	0.2911	0.2631	0.0569	0.0625	0.0541	0.0582
KETOD	SimpleTOD	0.0821	0.1015	0.0910	0.0538	0.1147	0.1362	0.1268	0.0726	0.0178	0.0266	0.0149	0.0106
	SOLOIST	0.0970	0.1018	0.0945	0.0848	0.0957	0.1185	0.0933	0.0675	0.0167	0.0145	0.0174	0.0185
	InstructTODS	0.0671	0.0825	0.0640	0.0593	0.0934	0.1102	0.0897	0.0783	0.0105	0.0140	0.0100	0.0075
	SyncTOD	0.0488	0.0495	0.0510	0.0456	0.1005	0.0896	0.1185	0.0959	0.0068	0.0052	0.0086	0.0068
	Q-TOD	0.0159	0.0186	0.0151	0.0134	0.0917	0.1053	0.0968	0.0681	0.0073	0.0026	0.0101	0.0098
	ZS-ToD	0.0394	0.0439	0.0254	0.0385	0.0183	0.0231	0.0059	0.0250	0.0260	0.0328	0.0198	0.0243
	Auto-ToD	0.0480	0.0528	0.0492	0.0415	0.0797	0.0678	0.0932	0.0812	0.0134	0.0157	0.0151	0.0092
	GPT-2	0.1890	0.2106	0.1961	0.1524	0.2105	0.2437	0.2078	0.1687	0.0346	0.0500	0.0252	0.0263
	Llama-3.2	**0.2398**	**0.2864**	**0.2354**	**0.1862**	**0.2701**	**0.3165**	0.2579	0.2208	0.0581	0.0723	**0.0508**	**0.0490**
	FLAN-T5	0.2082	0.2351	0.2048	0.1792	0.2727	0.3025	**0.2811**	**0.2234**	0.0526	**0.0750**	0.0454	0.0339

Table 7: BLEU Scores for Overall Response Generation, Inform and Request.

by generally lower scores. Despite this, analyzing performance across these subtasks provides valuable insights into the system's capabilities, particularly in adapting to different dialogue roles and handling both information retrieval and user engagement.

We can see that ZeroToD models perform considerably better than baseline approaches across all tasks. Among the ZeroToD models, we can see that BLEU-4 scores are better for Llama-3.2 than FLAN-T5, particularly for the seen domains. Since the BLEU-4 metric is calculated by n-gram matches, Llama-3.2 having better supervised performance tends to generate responses closer to the ground truth, thus yielding higher BLEU-4 scores. Looking at the results, Llama-3.2 consistently outperforms other models across all response generation tasks, with this pattern being more pronounced in seen domains compared to unseen ones. This suggests that the model's larger capacity allows it to better learn and reproduce the linguistic patterns present in the training data.

The performance gap is most evident in the Inform subtask, where Llama-3.2 demonstrates superior ability to incorporate specific information into responses while maintaining natural language flow. This advantage likely stems from its enhanced ability to understand and integrate contextual information from the dialog history. For the Request subtask, which requires models to formulate appropriate questions, Llama-3.2 maintains its lead but with smaller margins, indicating that question formulation remains challenging even for larger models.

These patterns suggest that Llama-3.2's enhanced supervised learning capabilities particularly benefit tasks requiring precise information integration, while more complex dialogue behaviors like question formulation remain challenging across model sizes.

F Domain Specific Results

To get a deeper understanding of the performance of Auto-ToD and ZeroToD models on unseen domains, we present domain specific results for the API Invoke Accuracy and Complete API Accuracy metrics in Table 8. For the API Invoke Accuracy, we observe the same pattern as before, with Auto-ToD having much lower scores than ZeroToD models across unseen domains.

From these results, we can make another interesting observation: Auto-ToD has higher Complete API Accuracy for simple unseen domains like `Alarm` and `Movies`, however it shows poor performance for complex unseen domains like `Restaurants`, `Buses`, and `Music`.

The fine-tuned ZeroToD models demonstrate better understanding of complex domain structures in unseen scenarios. The models maintain consistent performance across unseen domains, indicating the effectiveness of fine-tuning in achieving robust out-of-domain generalization. The larger models, FLAN-T5 and Llama-3.2, exhibit more stable performance across unseen domains compared to the smaller

Dataset	Domains	Complete API Accuracy			
		Auto-ToD	**GPT-2**	**Llama-3.2**	**FLAN-T5**
SGD	Alarm_1	76.67	15.56	**78.89**	61.11
	Buses_3	37.16	29.05	46.62	**57.43**
	Events_3	27.78	55.56	50.79	**60.32**
	Homes_2	39.58	69.44	**76.39**	74.31
	Hotels_4	40.29	49.64	**100.00**	77.70
	Movies_3	47.46	23.73	**77.97**	67.80
	Music_3	30.19	58.49	73.58	**84.91**
	RentalCars_3	41.59	35.40	54.87	**63.72**
	Restaurants_2	28.68	84.56	77.94	**85.29**
KETOD	Alarm_1	66.67	00.00	**100.00**	66.67
	Buses_3	09.09	18.18	**45.45**	36.36
	Events_3	45.45	**72.73**	**72.73**	54.55
	Homes_2	**70.59**	41.18	64.71	**70.59**
	Hotels_4	37.50	37.50	81.25	**100.00**
	Movies_3	**57.14**	28.57	**57.14**	**57.14**
	Music_3	22.22	44.44	**88.89**	66.67
	RentalCars_3	**56.25**	31.25	50.00	50.00
	Restaurants_2	25.00	00.00	**87.50**	62.50

Table 8: Domain-wise evaluation of Complete API on unseen domains for Auto-ToD and ZeroToD models (RQ3).

GPT-2 model.

G Dialog Examples

Table 9 shows an example dialog in the Restaurant domain. The table contains the turn id, user utterance, gold response, SOLOIST, Auto-ToD, GPT-2, Llama-3.2, and FLAN-T5 response. Text highlighted in red outlines the portions where the system response by a model is incorrect and green highlights the correct parts. Texts highlighted in orange indicate that the model is partially correct and is missing some information.

In the example dialog, we can see that the responses of Auto-ToD is longer and more descriptive, whereas ZeroToD models produce succinct responses, similar to the gold standards. This observation indicates that ZeroToD was able to pick up the text generation style of the gold standard, which is an important aspect for ToD systems.

The baseline SOLOIST fails to generate a valid response on a few turns. This stems from the complex structure of the output, which consists of dialog states, actions and system response. The output structure has a certain format and different parts are identified with special tokens. On many occasions, the model does not produce the correct tokens, thus leading to invalid generations.

On turns where the user asks about multiple information, the GPT-2 and SOLOIST model struggles to respond to all the user requests. However, all the other models do not have this problem. This could be attributed to the fact that GPT-2 models are the smallest model when compared to all the other models.

Llama-3.2 and FLAN-T5 generates responses that are very similar to the gold standard. These models are able to pickup the generation style and can respond to multiple requests of the user. In the example dialog, the user provided information about the restaurant name, time and date on turn 1, but then changed the restaurant name on turn 2. Llama-3.2 and FLAN-T5 models were able to identify what changed and moved forward with the API Call by using all the correct information. However, Auto-ToD was not able to keep track of all the changes and made an incorrect API Call. SOLOIST was able to make a partially correct API Call since it was not able to generalize to the `number_of_seats` slots. It used the `party_size` slot that it had seen during training.

This dialog example showed some of the drawbacks of baseline approaches and some example cases of where they struggle. For ZeroToD models, the limitations of GPT-2 was evident, as it was not able to process multiple requests of the user. However, Llama-3.2 and FLAN-T5 had performance similar to the gold standard across multiple turns, which falls in line to our automatic metrics and human evaluation.

Table 9: Dialog Example—SGD dataset. Dialog id 1_00001

Model	**Response—SGD dataset. Dialog id 1_00001**
Turn	1
User	Can you book a table for me at the Ancient Szechuan for the 11th of this month at 11:30 am?
Gold	In which city are you trying to book the table?
SOLOIST	Please confirm the following details: Booking a table for 2 at Ancient Szechuan in San Francisco on March 11th at 11:30 am.
Auto-ToD	Could you please provide the city where the restaurant 'Ancient Szechuan" is located and the number of seats you would like to reserve?
GPT-2	Sure, I have a table for 2 at Ancient Szechuan in San Francisco at 11:30 am today.
Llama-3.2	What is the location of the restaurant?
FLAN-T5	Where should I look for the restaurant?
Turn	2
User	Can you book a table at the Butterfly restaurant in San Francisco?
Gold	Please confirm that you need a table for 2 at the Butterfly Restaurant in San Francisco at 11:30 am on March 11th.
SOLOIST	Can you please confirm that you want to book a table for 2 at the Butterfly restaurant in San Francisco at 11:30 am on March 11th?
Auto-ToD	Could you please confirm the date and time for the reservation at the Butterfly restaurant in San Francisco? Additionally, let me know the number of seats you would like to reserve.
GPT-2	Can you please confirm that you want to book a table for 2 people at the Butterfly restaurant in San Francisco at 11:30 am on March 11th?
Llama-3.2	Can you confirm the following details: You want to book a table at the butterfly Hotel in San Francisco at 11:30 am. You want 2 seats. You want to book the table on March 11th.
FLAN-T5	You wish to book a table for 2 at the Butterfly Restaurant in San Francisco for March 11th at 11:30 am. Is this correct?
Turn	3
Gold	ApiCall(method='ReserveRestaurant', parameters= 'date': '2019-03-11', 'location': 'San Francisco', 'number_of_seats': '2','restaurant_name': 'Butterfly Restaurant', 'time': '11:30')
SOLOIST	ApiCall(method='ReserveRestaurant', parameters='city': 'San Francisco', 'date': '2019-03-11', 'party_size': '2','restaurant_name': 'The Butterfly Restaurant', 'time': '11:30')
Auto-ToD	ApiCall(method='FindRestaurants',parameters='category': 'Butterfly', 'location': 'San Francisco')
GPT-2	ApiCall(method='ReserveRestaurant', parameters='date': '2019-03-11', 'location': 'San Francisco', 'number_of_seats': '2','restaurant_name': 'The Butterfly Restaurant', 'time': '11:30')
Llama-3.2	ApiCall(method='ReserveRestaurant', parameters='date': '2019-03-11', 'location': 'San Francisco', 'number_of_seats': '2','restaurant_name': 'Butterfly Restaurant', 'time': '11:30')
FLAN-T5	ApiCall(method='ReserveRestaurant', parameters= 'date': '2019-03-11', 'location': 'San Francisco','restaurant_name': 'Butterfly Restaurant', 'number_of_seats': '2', 'time': '11:30')
Continued on next page...	

Dataset	Response—SGD dataset. Dialog id 1_00001
Turn	4
User	That's perfect. What cuisine do they offer?
Gold	Your table has been booked successfully and they serve Asian cuisine.
SOLOIST	Your table has been booked.
Auto-ToD	It looks like the reservation at the Butterfly Restaurant in San Francisco has been successfully made for March 11th at 11:30 am for 2 people. The restaurant offers Asian cuisine. If you need any more assistance or have any other questions, feel free to ask. Have a great day! Goodbye!
GPT-2	Your table is booked.
Llama-3.2	Your reservation has been successful. They offer Asian cuisine.
FLAN-T5	Your reservation has been made. They serve Asian food.
Turn	5
User	Can you get me the user rating of the restaurant? Is it expensive?
Gold	Their prices are moderate and they have a user rating of 4.0.
SOLOIST	✗ (Failed to generate a valid response)
Auto-ToD	The Butterfly Restaurant in San Francisco has a user rating of 4.00. The price range is moderate, so it's not considered expensive. If you have any more questions or need further assistance, feel free to ask!
GPT-2	The rating is 4.0.
Llama-3.2	The restaurant has a 4.0 rating and is considered moderate in price.
FLAN-T5	They have a moderate price range and a 4.0 rating.
Turn	6
User	Thanks a bunch!
Gold	Is there anything else I can do for you?
SOLOIST	✗ (Failed to generate a valid response)
Auto-ToD	You're welcome! If you need any more assistance, feel free to ask. Have a great day! Goodbye!
GPT-2	Is there anything else I can help you with?
Llama-3.2	Do you need anything else?
FLAN-T5	Do you need help with anything else?
Turn	7
User	No. Thank you. That's all I need.
Gold	Have a great day!
SOLOIST	Have a great time at the restaurant!
Auto-ToD	You're welcome! If you need any more assistance in the future, feel free to reach out. Have a great day! Goodbye!
GPT-2	Have a good one!
Llama-3.2	Have a great day.
FLAN-T5	Have a nice day.

H 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.

Thank you for your time and valuable feedback!