TOD-Flow: Modeling the Structure of Task-Oriented Dialogues

Sungryull Sohn[1*] **Yiwei Lyu**[2*] **Anthony Zhe Liu**[2] **Lajanugen Logeswaran**[1]
Dong-Ki Kim[1] **Dongsub Shim**[1] **Honglak Lee**[1,2]
[1]LG AI Research [2]University of Michigan, Ann Arbor

Abstract

Task-Oriented Dialogue (TOD) systems have become crucial components in interactive artificial intelligence applications. While recent advances have capitalized on pre-trained language models (PLMs), they exhibit limitations regarding transparency and controllability. To address these challenges, we propose a novel approach focusing on inferring the TOD-Flow graph from dialogue data annotated with dialog acts, uncovering the underlying task structure in the form of a graph. The inferred TOD-Flow graph can be easily integrated with any dialogue model to improve its prediction performance, transparency, and controllability. Our TOD-Flow graph learns what a model can, should, and should not predict, effectively reducing the search space and providing a rationale for the model's prediction. We show that the proposed TOD-Flow graph better resembles human-annotated graphs compared to prior approaches. Furthermore, when combined with several dialogue policies and end-to-end dialogue models, we demonstrate that our approach significantly improves dialog act classification and end-to-end response generation performance in the MultiWOZ and SGD benchmarks. Code available at: `https://github.com/srsohn/TOD-Flow`

1 Introduction

Task-Oriented Dialogue (TOD) systems have attracted significant attention due to their potential applications in personal assistants, customer support, and other interactive systems that necessitate human-like conversation [2, 39]. Many of the recent advances in TOD have heavily leaned on pre-trained language models (PLMs) [12, 38] that are first pre-trained on a large corpus of data in an unsupervised manner, and then either fine-tuned [12, 6, 36] or subjected to few-shot prompting [15, 17] to adapt them to specific dialogue domains. While these approaches have yielded commendable performance, they have limitations. Few-shot prompted models have been challenged by issues of transparency, controllability, and adaptability to specific domains, especially when working with only a few examples. The lack of understanding of their decision-making processes and fine-grained control over their output is often inadequate, which can result in sub-optimal conversational experiences. On the other hand, fine-tuned models are confronted with their own unique challenges. While they offer improved performance by aligning the model with task-specific semantics, this approach typically requires large annotated datasets and resources which can be a limiting factor in practice. Furthermore, these models often lack transparency, making it challenging to understand the reasons behind their decisions or predictions.

Some prior works [29, 18] introduced workflow-based dialog models to handle the challenges in existing TOD models. These methods aim to explicitly model the structure of dialog in a graph format. Grounding the dialog in the graph offers benefits in terms of 1) elucidating the reasoning of

[*]Equal Contribution

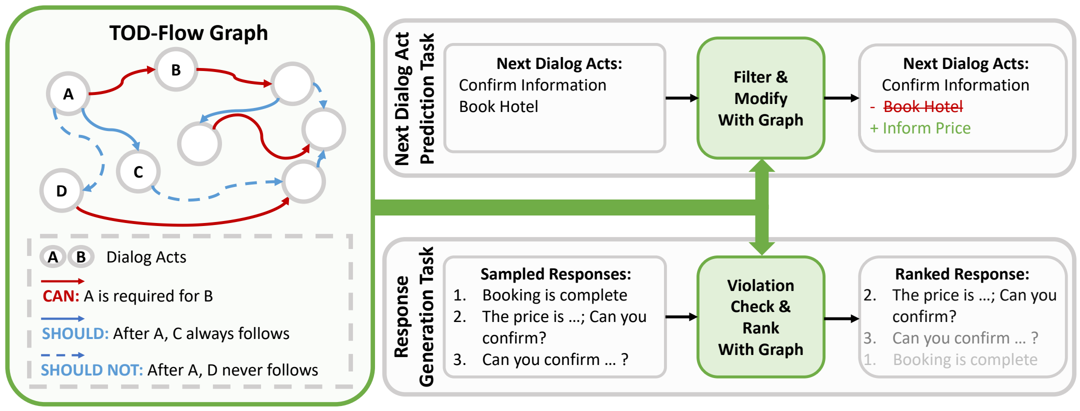


Figure 1: (Left) Our TOD-Flow graph captures the causal dependency between dialog acts in terms of can, should, and should not relationships. (Right) Intuitively, given the TOD-Flow graph, we can predict the relevant and irrelevant dialog act or responses based on the current dialog state. Based on the predicted relevance, we can filter and rank the base model outputs to enhance the prediction performance in both dialog act classification and end-to-end response generation tasks.

system's decisions in terms of the relationships (*i.e.*, transparency); 2) allow human manipulation of the dialogue model via graph modification (*i.e.*, controllability) without retraining the dialog model. However, real-world dialogues were often unstructured, making it non-trivial to be modeled as a workflow, and the necessity of manually designing the domain-specific workflow or its elements limits its practical applicability.

To tackle these challenges, we propose to learn the *subtask graph* [33] from task-oriented dialog data. Intuitively, the subtask graph can predict the affordance (*i.e.*, availability) of the action from the status of environment and agent (*i.e.*, the progress of completing a task or the subtasks). The subtask graph framework has two major benefits: i) subtask graphs can be inferred from the demonstrations without any direct supervision (*e.g.*, video [16] or transcript [21]), ii) subtask graphs can be combined with the base prediction model to improve its prediction since the subtask graph does not decide what to predict but instead suggests the affordable candidates of prediction.

Contributions. The main contribution of this work is generalizing the subtask graph framework into task-oriented dialog settings. To this end, we propose the *TOD-Flow graph*, which extends the subtask graph framework in three major aspects. First, we show that subtask graph can infer the relationship between dialog state and dialog acts without requiring any manual definition of nodes and edges in graphs. Second, in addition to the precondition (or *can* relationship), we present learning algorithms to model two novel relationships, *should* and *should not*, which provide more fine-grained control and improved prediction. For instance, a *can* relationship may represent that the system can make a payment only if the user confirms the payment. The *should* relationship may learn that if a user ask about the address of the hotel, the system should reply back. Conversely, a *should not* relationship may dictate that the system usually does not predict a farewell if the user's last utterance implies a question. Third, we demonstrate that the inferred TOD-Flow graphs can enhance any dialog policy or end-to-end dialog system, whether fine-tuned or prompted, without the necessity of retraining.

2 Background

Our main contribution, the TOD-Flow graph, is an extension of the subtask graph framework [33, 34], which describes the causal dependency structure of a compositional task τ consisting of a set of subtasks. In the context of task-oriented dialogue, dialog acts can be seen as subtasks. Each subtask has a **precondition** that must be satisfied before the subtask can be performed. Note that the precondition is not the only relationship between subtask, and in Section 3.2 we extend it by incorporating other types of relationships. Since precondition describes the causal relationship between subtasks, it imposes a constraint on the order in which subtasks can be performed (*e.g.*, the system can make a payment *only after* the user confirms the payment). Formally, we define the precondition as a Boolean expression consisting of Boolean constants (*e.g.*, True or False), Boolean variables and logical connectives (*e.g.*, and (&), or (|)). To illustrate, consider the precondition of subtask C: $f_{\mathsf{C}} = \&(\mathsf{A}, \mathsf{B})$, where the subtasks A and B must be completed before C is completed. It can be equivalently viewed as a Boolean function where inputs are Boolean variables indicating whether subtasks A and B are completed, and the output represents whether the precondition f_{C} is satisfied: $f_{\mathsf{C}}(\mathsf{A} = \text{True}, \mathsf{B} = \text{False}) = \text{True} \ \& \ \text{False} = \text{False}$. Also, the boolean expression $f_{\mathsf{C}} = \&(\mathsf{A}, \mathsf{B})$ can be viewed as a **graph** with vertices consisting of subtasks and logical operators $V = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \&\}$ and edges $E = \{\mathsf{A} \to \&, \mathsf{B} \to \&, \& \to \mathsf{C}\}$ that represent preconditions. We will use these different views of the precondition (*i.e.*, as a boolean expression, graph or function) interchangeably. The **subtask graph** visualizes the preconditions $f_1, \ldots$ of the subtasks (see Figure 1 for examples). We note that the subtask graph has been adopted in various settings [20, 34, 35] and subsumes other task graph formats [1, 3, 32], flowchart [29], and workflow [18].

3 TOD-Flow Graph Learning

3.1 Problem Formulation

For dialogue turn t, let u_t be the user input and and r_t be the corresponding system response. The user inputs and system responses are represented as a set of *dialog acts* that labels a piece of dialog according to its category of meaning: $u_t, r_t \in \mathcal{A}$, where $\mathcal{A}$ is the set of dialog acts. Let d_t be the database query result that can be obtained through querying the database. Then, the dialog data $\mathcal{D}$ is a set of dialog trajectories $\mathcal{D}^\tau = \{(u_0, r_0, d_0, u_1, \ldots), \ldots\}$. Given the dialog data $\mathcal{D}^\tau$, the goal of *TOD-flow graph learning* task is to generate the *TOD-flow graph G* that models the dependency between system acts, user acts, and database results in graph format.

Challenges. There are two main challenges in tackling this task. First, the information in the dialogue is noisy due to annotation errors such as missing or ambiguous dialog acts, slots, and values. Second, these dialog annotations provide only the partial information about underlying relationships between subtasks. Thus, we need to infer whether each relationship is satisfied or not from the dialogue annotations. We describe how we overcome these challenges in Section 3.3.

3.2 TOD-Flow Graph

For each dialog act $\mathbf{a}$, the TOD-flow graph is defined in terms of three conditions: $\mathsf{Can}_{\mathbf{a}}$, $\mathsf{Shd}_{\mathbf{a}}$, and $\mathsf{Shdnt}_{\mathbf{a}}$. Intuitively, $\mathsf{Can}_{\mathbf{a}}$, $\mathsf{Shd}_{\mathbf{a}}$ and $\mathsf{Shdnt}_{\mathbf{a}}$ condition respectively defines whether the dialog act $\mathbf{a}$ *can*, *should*, and *should not* be performed by the agent (user or system) at a given status. Similar to the precondition in subtask graph framework (see Section 2), each condition is defined as a Boolean expression. Also, it can be equivalently viewed as a Boolean function $f^{\mathsf{Can}}, f^{\mathsf{Shd}}, f^{\mathsf{Shdnt}} : \mathbf{c} \mapsto \{0, 1\}$ or a graph (see Section 2), where $\mathbf{c} \in \{0, 1\}^{N_\tau}$ is the subtask completion (or dialog state) vector indicating whether n^{th} subtask has been achieved (*i.e.*, $c[n] = 1$) or not (*i.e.*, $c[n] = 0$).

3.3 Learning TOD-Flow Graphs

Dataset Given the dialogue data $\mathcal{D} = \{(u_t, d_t, r_t)\}$, we aim to build the graph inference dataset $\mathcal{D}_G = \{(\mathbf{c}_t, \mathbf{a}_t)\}$, from which we can infer the TOD-flow graph f^{Can}, f^{Shd}, and f^{Shdnt}. The action set $\mathbf{a}_t$ is the set of dialog acts that were performed at turn t: $\mathbf{a}_t = u_t \cup r_t$. The completion set $\mathbf{c}_t$ is the set of dialog acts and database query that has ever been performed before turn t: $\mathbf{c}_t = \mathbf{c}_{t-1} \cup \mathbf{a}_{t-1} \cup \{d_{t-1}\}$.

Shd inference. When should condition is satisfied (*i.e.*, $f_n^{\mathsf{Shd}}(\mathbf{c})$=1), the agent is required to perform the n^{th} dialog act (*i.e.*, $a[n] = 1$). When the should condition is not satisfied (*i.e.*, $f_n^{\mathsf{Shd}}(\mathbf{c})$=0), the should relationship has no effect on the policy. This relationship can be represented as the confusion matrix shown in the first two rows in Table 1. Accordingly, we maximize the true positive, while minimizing the false positive by maximizing the objective J_{Shd} in Equation (1).

$$J_{\mathsf{Shd}} = \mathbb{E}_{(\mathbf{c}, a[n])} \left[\mathbb{I}(a[n] = 1 | f_n^{\mathsf{Shd}}(\mathbf{c}) = 1) \right] \tag{1}$$

Cond \ $a[n]$	Executed ($a[n]=1$)	Not executed ($a[n]=0$)
$f_n^{\mathsf{Shd}}=1$	True positive	False positive
$f_n^{\mathsf{Shd}}=0$	-	-
$f_n^{\mathsf{Shdnt}}=1$	False positive	True positive
$f_n^{\mathsf{Shdnt}}=0$	-	-
$f_n^{\mathsf{Can}}=1$	True positive	-
$f_n^{\mathsf{Can}}=0$	False negative	-
$f_n^{\mathrm{BC}}=1$	True positive	False positive
$f_n^{\mathrm{BC}}=0$	False negative	True negative

Table 1: Confusion matrix of the TOD-flow graphs (Shd, Shdnt, Can) and the BC baseline with respect to the dialog act label $a[n]$. The empty cell ($-$) indicates that the label for each relationship is unavailable.

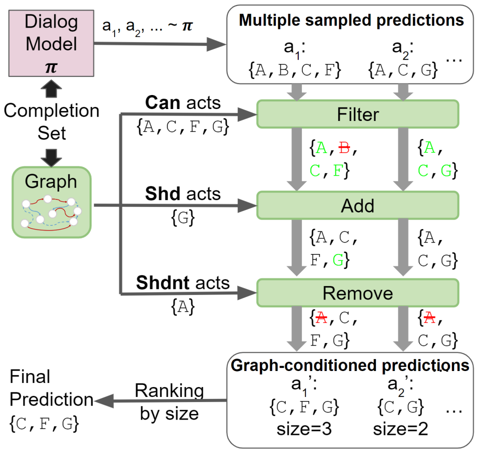


Figure 2: An illustration of our graph-conditioned dialog policy. We sample multiple predictions from base dialog policy π, then use our graph to filter, add and remove acts from each prediction following Algorithm 1. Then we obtain final prediction by selecting the graph-conditioned prediction with the largest size.

Shdnt inference. Similar to Shd, when the shoud-not condition is satisfied, (*i.e.*, $f_n^{\mathsf{Shdnt}}(\mathbf{c})=1$), the agent is required *not* to perform the n^{th} dialog act (*i.e.*, $a[n]=0$), and agent has a freedom to execute the n^{th} dialog act when the condition is not satisfied (*i.e.*, $f_n^{\mathsf{Shdnt}}(\mathbf{c})=0$). Table 1 summarizes the relationship between Shdnt and $a[n]$. We learn f_n^{Shdnt} by maximizing the following objective:

$$J_{\mathsf{Shdnt}} = \mathbb{E}_{(\mathbf{c},a[n])}\left[\mathbb{I}(a[n]=0|f_n^{\mathsf{Shdnt}}(\mathbf{c})=1)\right]. \tag{2}$$

Can inference. By definition of Can (or precondition), a dialog act can only be performed ($\mathbf{a}_n=1$) if its Can condition is satisfied (*i.e.*, $f_n^{\mathsf{Can}}(\mathbf{c})=1$): a true positive case in Table 1. On the contrary, it is a contradiction if a dialog act $\mathbf{a}_n$ is performed while its Can is not satisfied (*i.e.*, $f_n^{\mathsf{Can}}(\mathbf{c})=0$): a false negative case in Table 1. Thus, the f^{Can} can be learned by maximizing the following objective:

$$J_{\mathsf{Can}} = \mathbb{E}_{(\mathbf{c},a[n])}\left[\mathbb{I}[f_n^{\mathsf{Can}}(\mathbf{c})=1|a[n]=1]\right]. \tag{3}$$

However, different from Shd and Shdnt, inferring Can is nontrivial, because if we maximize J_{Can}, we get the trivial precondition: always true (*i.e.*, $f_n^{\mathsf{Can}}(\mathbf{c})=1$ for all $\mathbf{c}$). Previous works handled this issue by either making additional assumptions [10, 13] or applying regularization [16]. However, these approaches unavoidably introduces noise in learning, and require careful hyperparameter tuning to balance between objective and regularization. Instead, inspired by the fact that Can and $\neg$Shdnt (*i.e.*, negation of Shdnt) applies to the policy in the same manner (*i.e.*, the dialog act $\mathbf{a}$ can be performed if both $f_n^{\mathsf{Can}}=1$ and $f_n^{\neg\mathsf{Shdnt}}=1$), we propose to infer Can and Shdnt simultaneously as follows:

$$\begin{aligned} J_{\mathsf{Can}\wedge\neg\mathsf{Shdnt}} = \mathbb{E}_{(\mathbf{c},a[n])} & \Big[\mathbb{I}[f_n^{\mathsf{Can}\wedge\neg\mathsf{Shdnt}}(\mathbf{c})=1|a[n]=1] \\ & + \alpha\,\mathbb{I}[f_n^{\mathsf{Can}\wedge\neg\mathsf{Shdnt}}(\mathbf{c})=0|a[n]=0]\Big], \end{aligned} \tag{4}$$

where α determines the relative weight between Can and Shdnt in optimization. Intuitively, the agent can perform n^{th} dialog act only if Can is satisfied and Shdnt is not satisfied.

Baseline As an ablation model, we consider the behavioral cloning (BC) [24] objective, which tries to mimic the demonstration behavior:

$$J_{\mathrm{BC}} = \mathbb{E}_{(\mathbf{c},a[n])}\left[\mathbb{I}\left[f_n^{\mathrm{BC}}(\mathbf{c})=a[n]\right]\right] \tag{5}$$

The confusion matrix is shown at the bottom of Table 1.

We can use any binary classification models to optimize the objectives (1), (4), and (5). We used the decision tree models in the experiment following the previous works [3, 14, 34].

Algorithm 1 TOD-flow graph-conditioned dialogue model

```
Require: Dialogue model $\pi$, TOD-flow graph $f^{\mathtt{Can}}$, $f^{\mathtt{Shd}}$, $f^{\mathtt{Shdnt}}$, Completion $\mathbf{c}$
Ensure: Sampled dialog acts $\mathbf{a}$
1: $\mathbf{a} \sim \pi$                                                        ▷ Sample dialog acts from $\pi$
2: $\mathbf{a} \leftarrow \mathbf{a} \cup \{\mathbf{a}' | f^{\mathtt{Shd}}_{\mathbf{a}'}(\mathbf{c}) = 1\}$                            ▷ Apply Shd
3: $\mathbf{a} \leftarrow \mathbf{a} \cap \{\mathbf{a}' | f^{\mathtt{Can} \wedge \neg \mathtt{Shdnt}}_{\mathbf{a}'}(\mathbf{c}) = 1\}$           ▷ Apply Can ∧ ¬Shdnt
4: return $\mathbf{a}$
```

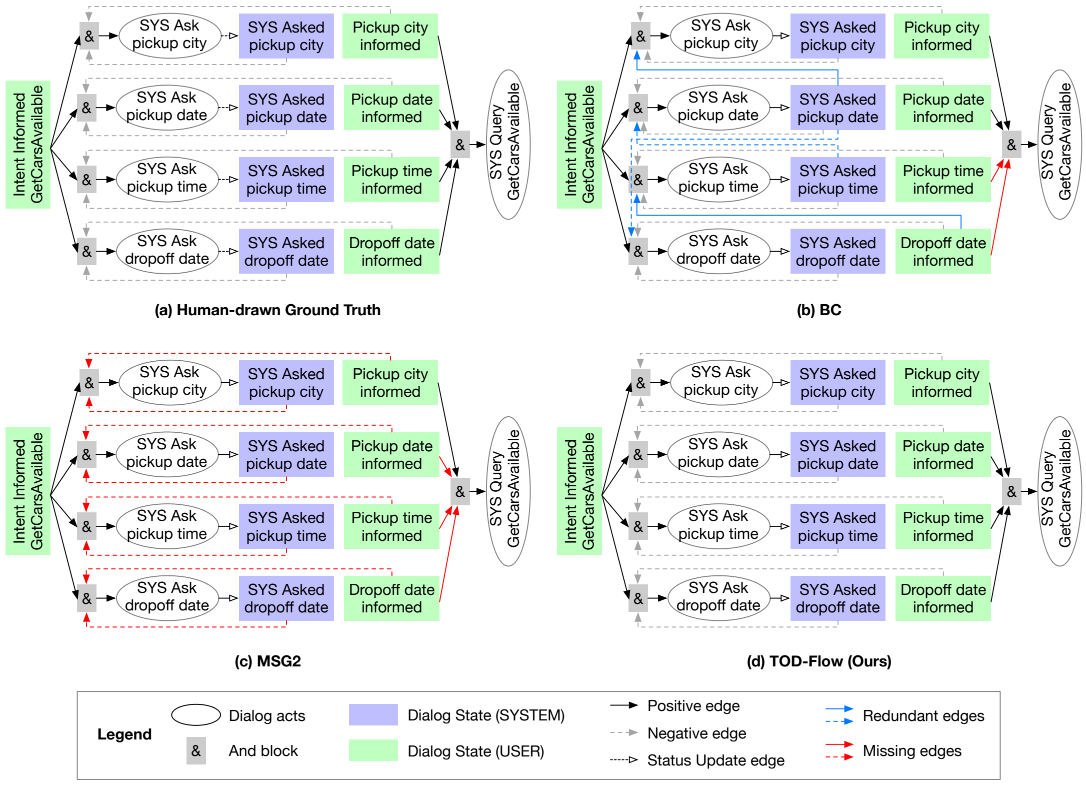


Figure 3: Comparing baselines and our method against human-drawn ground truth graph on a subpart of the `RentalCars_1` domain of SGD. Dotted lines are negative edges (logical negations). Red Edges are missing edges and blue edges are redundant edges compared to the ground truth. In this scenario, the system is allowed to ask for a slot if user has informed the intent of finding an available car and has not yet informed this slot and the system have not yet asked for the slot; and the system can only query for a car when all 4 slots are informed. Our method perfectly matches the ground truth, while both baselines have incorrect edges (**BC** has 4 redundant edges and 2 missing edges, while **MSG**^{2} has 11 missing edges).

4 Graph-conditioned Dialog Modeling

We describe how the inferred TOD-flow graph G can enhance the prediction performance of any off-the-shelf dialogue policies and end-to-end dialog systems.

4.1 Graph-conditioned Dialog Policy

The inferred TOD-flow graph (Shd and Can ∧ ¬Shdnt) can propose the dialog acts that can, should, and should not be performed given the current dialog history (or completion set). Figure 2 and Algorithm 1 describes the entire process. Given a base dialogue policy π^{DP}, we sample the system acts from the policy $a \sim \pi^{\text{DP}}$, and filter, add and remove acts from the sampled system acts according to the Can, Shd, and Shdnt, respectively.

Models	SGD (24 domains)		MultiWOZ (14 domains)	
	FLAN-T5	GPT-turbo	FLAN-T5	GPT-turbo
No Graph	49.9%	78.8%	21.6%	40.8%
+**BC**	57.2%	71.8%	23.5%	38.2%
+**MSG**^{2} [16]	52.4%	79.4%	23.8%	40.2%
+**TOD-Flow** (ours)	**83.1%**	**89.2%**	**35.0%**	**48.2%**

Table 2: Average F-1 scores of next system action prediction experiment on two datasets (SGD and MultiWOZ) and two large language models (FLAN-T5 and GPT-turbo) as few-shot predictors. We can see that while **BC** often damages performance, **TOD-Flow** consistently improves performance by a significant amount.

We can further improve the prediction performance if our baseline dialog model can be sampled multiple times with different results, as illustrated in Figure 2. We use the graph to condition each candidate result, then select the best one using a selection method such as most number of actions in set, candidate with least graph violations, etc. We empirically found that simply choosing the result with the most actions works best.

4.2 Graph-conditioned End-to-end Response Generation

End-to-end dialogue system directly reads and outputs the utterances in natural language form. Given the base end-to-end model π^{e2e}, we sample multiple system utterances from the base model (mostly via beam-search alternates, see Appendix A.2.1 for details). Then, we use a few-shot prompted GPT-turbo model to annotate each generated candidate utterance with the dialog act (see Appendix A.2.3 for details). Finally, we use the inferred graph to choose the best utterance that has the least *violation rate* (*i.e.*, portion of dialog acts that violates the inferred `Can`, `Shd`, and `Shdnt` conditions). Note that with this approach, all final responses still comes from the base end-to-end model, so the improvement is still upper-bounded by the capabilities of the model. Our graph simply presents a better candidate selection method.

5 Experiments

We perform experiments to show that (1) the TOD-flow graph can be accurately predicted without any supervision, (2) our graph can improve the accuracy of dialog policy models, and (3) our graph can improve the quality of response generation in end-to-end dialog models.

5.1 Dataset

We used two standard TOD benchmarks. Schema-Guided Dialogue (SGD) [31] has over 20k task-oriented simulated conversations based on human-designed schema. SGD covers a wide range of domains (*i.e.*, different dialog acts and goals). We use 24 domains in SGD, and did not use the schema for experiment. MultiWOZ [4] has 10k human-human conversations on 14 domains. Since MultiWOZ is collected from human-human conversations, the utterances are much more linguistically diverse than SGD. Also, different from SGD where the annotations are generated from the schema, annotations in MultiWOZ are labeled by human. Therefore, the annotations in MultiWOZ are often noisy (*i.e.*, inconsistent, wrong, or missing), which present additional challenge compared to SGD.

For both datasets, we obtain train/test splits of the dialogs within each domain (see Appendix A.1.1 for details). The training set is used for 1) inferring **TOD-Flow** graph, 2) building demonstration for few-shot prompted models, and 3) finetuning the finetuning-based models. The test set is only used for evaluation. For graph inference, we map the dialog act of user, database, and system to completion $\mathbf{c}$ and dialog act $\mathbf{a}$ vectors as described in Section 3.3.

5.2 Baselines

We compare three graph inference algorithms:

	`Can-Shdnt` graph	`Shd` graph	Ranking method	SGD (24 domains)		MultiWOZ (14 domains)	
				FLAN-T5	GPT-turbo	FLAN-T5	GPT-turbo
No graph	✗	✗	Greedy	49.9%	78.8%	21.6%	40.8%
Graph ablations	✗	✓	Compliance	70.4%	82.5%	26.1%	41.6%
	✓	✗	Compliance	65.6%	83.5%	30.4%	44.5%
Sampling & ranking ablations	✓	✓	Greedy	80.9%	88.5%	26.2%	45.5%
	✓	✓	Majority	75.2%	88.1%	20.7%	45.0%
	✓	✓	Violation	76.2%	88.3%	23.7%	44.7%
	✓	✓	Uniform	75.8%	88.0%	23.2%	45.2%
Ours	✓	✓	Compliance	**83.1%**	**89.2%**	**35.0%**	**48.2%**

Table 3: Ablation studies on graphs and ranking methods for next system action prediction. The numbers shown are average F-1 scores. We see that both `Can-Shdnt` and `Shd` graphs have significant contributions towards the performance and our Compliance ranking method outperforms greedy sampling or other ranking methods.

- **BC** learns to imitate the demonstration via behavioral cloning (see Section 3.3)
- **MSG**^{2} [16] learns the subtask graph by optimizing the $J_{\texttt{Can}}$ (see Equation (3)) with complexity regularization.
- **TOD-Flow** (ours) is our **TOD-Flow** graph learning algorithm.

For fair comparison, we used the scikit-learn decision tree model [27] for all the graph inference algorithms.

5.3 TOD-Flow Graph Inference

We first qualitatively compare the inferred graphs with the human-drawn graphs on `RentalCars_1` domain in SGD dataset. We found that in general **TOD-Flow** produces graphs that agree with the human-drawn graphs much more often compared to baselines (**BC** and **MSG**^{2}). Figure 3 illustrates the subpart of the inferred and human-drawn graph, where **TOD-Flow** inferred the graph perfectly matches the human-drawn graph, while the baselines missed important information (such as not requiring all 4 required slots to be informed before performing the query).

5.4 Task 1: Dialog Policy Learning

Base Models. We use two instruction-tuned large language models (LLM) as baseline dialog policy: FLAN-T5-xxl [7] and GPT-turbo[1]. At each turn, we prompt the LLM with five demonstration dialogues from train split of the same domain followed by the dialogue history, and ask the model to predict next system dialog acts. See Appendix A.1.2 for more details on prompting the LLM.

Evaluation Protocol. We sample 10 candidate predictions from the base models, which is filtered and ranked based on the graph (see Section 4.1) to choose the best prediction.

Metric. We measure F-1 score between the ground-truth and predicted system dialog acts at each turn and average over entire domains.

Results. Table 2 summarizes the F-1 score of each model on SGD and MultiWOZ. Overall, we observe that **TOD-Flow** consistently improves the prediction accuracy with a significant margin compared to other baselines **BC** and **MSG**^{2} on all base models and all dataset. We also found that the improvements are bigger on FLAN-T5 compared to GPT-turbo. This indicates that the GPT-turbo already models the `Can`, `Shd`, and `Shdnt` to some extent, so that augmenting it with the graph provide less benefits. Since **BC** learns to mimic the exact behavior in demonstration, **BC** tends to dictate the base policy more aggressively and hurts the performance when combined with strong base model GPT-turbo. **MSG**^{2} correctly models the precondition of dialog acts, but provides less benefit

[1]https://platform.openai.com/docs/models/gpt-3-5

compared to **TOD-Flow** due to the conservative graph learning (*i.e.*, complexity regularization) and lacking the ability to model `Shd` and `Shdnt` relations.

Ablations. To further justify our design choices, we performed ablation studies on two key components of **TOD-Flow**: the graphs and the ranking method after graph-conditioning. For filtering graphs, we examined the effect of `Can-Shdnt` and `Shd` graphs. Regarding the ranking method, we compare the proposed ranking approach (*i.e.*, Compliance) against various alternatives:

- **Greedy** ranks by likelihood of base LLM.
- **Compliance** ranks predictions by larger number of actions complying with the graph (*i.e.* rank by size in Figure 2).
- **Majority** chooses the majority prediction among the multi-sampled predictions.
- **Violation** ranks predictions by least number of actions filtered, added, and removed in graph conditioning.
- **Uniform** randomly chooses one of the multi-sampled predictions.

The results are shown in Table 3, and **TOD-Flow** outperformed all ablations, showing the necessity of all graphs and ranking by largest set.

5.5 Task 2: End-to-end Response Generation

Base Models. We use the three SOTA end-to-end dialogue models finetuned on MultiWOZ: GALAXY [12], HDNO [36], and HDSA [6] as base models. Note that for GALAXY, since we were unable to reproduce the official prediction using the official repository, we report the result with both official prediction (GALAXY) and the greedy (*i.e.*, beam search with beam width=1) prediction we obtained by running the official repository (GALAXY*).

Evaluation protocol. From each model, we first sample five system response utterances: one from official prediction[2] [25] and four from the model downloaded from the official implementation (see appendix A.2.1 for details). The graph conditioning process follows Section 4.2. As an ablation, we also evaluated our method without conditioning on `Shd` graphs.

Metric. We follow the standard evaluation metric using the official code [25], which computes three metrics on the MultiWOZ test set: *BLEU* (average *BLEU* [26] scores between generated and ground truth response), *Info* (percentage of dialogs where the system presents an appropriate entity) and *Succ* (percentage of dialogs where the task goals are achieved). The combined score is computed as $Score = BLEU + (Info + Succ)/2$. See Appendix A.2.2 for details on computing these metrics. We report all four metrics of the compared methods.

Result. We show the results in Table 4. We found that **TOD-Flow** can consistently improve *Info* and *Succ* metrics for all the base end-to-end dialog models. *BLEU* score fluctuates because it depends a lot on the exact wording of each response, which our graphs have no control over. The combined score consistently improves by 0.62, 0.6, 0.58, and 1.34 for HDSA, HDNO, GALAXY, and GALAXY*, respectively. Note that these score improvements are actually quite significant, as the difference between top and second top SOTA methods (GALAXY and HDNO) is only 0.74.

Next, we compare different ranking methods: **Uniform**, **BC**, **MSG^2**, and **TOD-Flow**. First, we consider **Uniform** method as a lower-bound performance which assign uniform ranking and randomly samples among the five sample predictions. Note that the performance of **Uniform** is significantly lower than the official prediction (*i.e.*, the first row of each base model). This is because the greedy prediction is in general much better in quality than the sampled predictions. Among the graph prediction methods, **TOD-Flow** consistently outperforms other baselines, **MSG^2** and **BC**. In fact, the **MSG^2** and **BC** often underperforms the official prediction. Since the sampled predictions are in general much worse than the greedy prediction, unless the graph-based ranking is highly accurate, it often samples the prediction that is worse than greedy prediction. On the contrary, **TOD-Flow** is often able to accurately pick out the non-greedy better alternative to outperform the official predictions. We show one such example in Table 5 in the Appendix.

[2]https://github.com/Tomiinek/MultiWOZ_Evaluation/tree/ master/predictions

	Graph	*BLEU*	*Info*	*Succ*	*Score*
HDSA	(no graph)	20.74	87.20	78.00	103.34
	+uniform	20.07	87.04	78.02	102.60
	+BC	**20.76**	86.80	77.70	103.01
	+MSG^2	20.71	86.90	77.90	103.11
	+TOD-Flow -No-Shd	20.69	87.70	78.20	103.64
	+TOD-Flow	20.51	**88.10**	**79.00**	**104.06**
HDNO	(no graph)	17.83	93.00	84.50	106.58
	+uniform	17.39	92.62	83.58	105.49
	+BC	17.79	92.90	84.50	106.49
	+MSG^2	17.83	92.90	84.40	106.48
	+TOD-Flow -No-Shd	**18.08**	93.10	**85.10**	**107.18**
	+TOD-Flow	17.97	**93.20**	85.00	107.07

	Graph	*BLEU*	*Info*	*Succ*	*Score*
GALAXY	(no graph)	**19.92**	92.00	82.80	107.32
	+uniform	18.28	88.94	80.38	102.94
	+BC	19.85	91.80	82.50	107.00
	+MSG^2	19.69	91.30	81.00	105.84
	+TOD-Flow -No-Shd	19.86	**92.40**	83.30	107.71
	+TOD-Flow	19.85	**92.40**	**83.70**	**107.90**
GALAXY*	(no graph)	18.88	90.70	80.70	104.58
	+uniform	17.86	87.86	79.54	101.56
	+BC	18.88	90.20	80.40	104.18
	+MSG^2	18.70	89.80	80.50	103.85
	+TOD-Flow -No-Shd	19.04	**91.10**	81.40	105.29
	+TOD-Flow	**19.12**	**91.10**	**82.30**	**105.82**

Table 4: Results from the response generation experiment with Galaxy [12], HDNO [36] and HDSA [6] as base models. The *Score* metric is computed by *Score* =*BLEU* +(*Info* +*Succ*)/2.

We also found that excluding Shd graphs significantly reduced the improvement in all base models except HDNO. This shows that, while using only Can-Shdnt graphs also consistently improves performance of all base models, including Shd graphs can generally get even better results.

6 Related Work

Task-oriented dialog systems are automated dialog systems that aims to help users achieve specific tasks, such as booking trains/hotels/restaurants, or providing information about weather/nearby attractions/nearby utilities. There are two general approaches to building these systems: pipeline approach and end-to-end approach [39]. The pipeline approach typically consists of several independent modules: (1) Natural Language Understanding (NLU): a module that translates input user utterance into standardized action and slot value representations. This task is often represented as a multi-class classification task and is usually done by sequence models such as RNNs [8, 9] and BERT [5]. Recent advances in large language models such as InstructGPT (Davinci series) and ChatGPT enables performing reliable NLU with in-context demonstrations [30]. (2) Dialog State Tracking (DST): a module that keeps track of the current dialog state (especially the slots and values). With a good NLU module, this step can usually be done in a rule-based fashion by accumulating user intents, actions and slot updates. Another approach is to use a language model to directly generate current dialog state [22]. (3) Dialog Policy: given the current dialog state, this module predicts the next system actions, usually via Reinforcement-Learning-based methods such as DQN/Policy Gradient [19, 40] or experience-replay [23]. (4) Natural Language Generation (NLG): given the current dialog states and the predicted next system actions, the NLG module needs to generate a response in natural language. This step is usually done through a fine-tuned language model [28, 22]. On the other hand, end-to-end task-oriented dialog systems does everything in one module. Some end-to-end approaches use one language model to perform all 4 steps [11], while other approaches skips some of the steps and directly generates final response [12, 37]. In general, the pipeline systems have the advantage of being more human-interpretable and each module can be updated or constrained independently (thus making them more controllable). Our work aims to improve the quality as well as controllability of the Dialog Policy module by automatically learning a precondition graph and constrain the predicted next actions with the graph, and we have also demonstrated that our method can improve end-to-end models as well.

7 Conclusion

This work has introduced a novel framework for improving the efficiency and predictive accuracy of task-oriented dialogues models. By leveraging the concept of subtask graph and generalizing it to a *TOD-flow graph*, we accurately inferred the latent task structure within a dialogue. As showcased through extensive experimentation with two public TOD datasets, the proposed technique has been proven to effectively generate accurate and human-interpretable graphs. Importantly, we have integrated these inferred graphs with a range of dialogue models, without necessitating retraining, resulting in a substantial enhancement in performance in both dialog act classification and end-to-end response generation.

Limitations

Although our method can be directly used when there are multiple domains involved in a single dialog (by treating combination of domains as a single separate domain and creating graphs using dialogs that has the same combination of domains, similar to what we did for MultiWOZ), this approach is limited in that (1) we need to infer a graph for every combination of domains that is present (such as the multi-domain dialogs in SGD, where there are 24 different single-domains alone), and (2) we cannot easily generalize to unseen domain combinations (even if we have graphs for each individual domain). In the future, we would like to explore ways to directly combine graphs for individual domains into multi-domain graphs and thus address the two limitations above.

We also relied on action annotations from the datasets to infer graphs, which limits the applicability of our approach. It would be interesting to extend our approach to unannotated raw dialogues.

References

[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. In *ICML*, 2017.

[2] Vevake Balaraman, Seyedmostafa Sheikhalishahi, and Bernardo Magnini. Recent neural methods on dialogue state tracking for task-oriented dialogue systems: A survey. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 239–251, Singapore and Online, July 2021. Association for Computational Linguistics.

[3] Craig Boutilier, Richard Dearden, Moisés Goldszmidt, et al. Exploiting Structure in Policy Construction. In *IJCAI*, 1995.

[4] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling, 2020.

[5] Qian Chen, Zhu Zhuo, and Wen Wang. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*, 2019.

[6] Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. Semantically conditioned dialog response generation via hierarchical disentangled self-attention, 2019.

[7] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

[8] Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE, 2014.

[9] Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719, 2016.

[10] Bradley Hayes and Brian Scassellati. Autonomously Constructing Hierarchical Task Networks for Planning and Human-Robot Collaboration. In *ICRA*, 2016.

[11] Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. Unified dialog model pre-training for task-oriented dialog understanding and generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 187–200, New York, NY, USA, 2022. Association for Computing Machinery.

[12] Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10749–10757, 2022.

[13] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration. In *CVPR*, 2019.

[14] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8565–8574, 2019.

[15] Vojtěch Hudeček and Ondřej Dušek. Are llms all you need for task-oriented dialogue? *arXiv preprint arXiv:2304.06556*, 2023.

[16] Yunseok Jang, Sungryull Sohn, Lajanugen Logeswaran, Tiange Luo, Moontae Lee, and Honglak Lee. Multimodal subtask graph generation from instructional videos. *arXiv preprint arXiv:2302.08672*, 2023.

[17] Tiziano Labruna, Sofia Brenna, Andrea Zaninello, and Bernardo Magnini. Unraveling chatgpt: A critical analysis of ai-generated goal-oriented dialogues and annotations, 2023.

[18] Issam H Laradji, Stefania Raimondo, David Vazquez, Pau Rodriguez, Christopher Pal, et al. Workflow discovery from dialogues in the low data regime. *Transactions on Machine Learning Research*, 2023.

[19] Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems, 2017.

[20] Anthony Z. Liu, Sungryull Sohn, Mahdi Qazwini, and Honglak Lee. Learning Parameterized Task Structure for Generalization to Unseen Entities. In *AAAI*, 2022.

[21] Lajanugen Logeswaran, Sungryull Sohn, Yunseok Jang, Moontae Lee, and Honglak Lee. Unsupervised task graph generation from instructional video transcripts. *arXiv preprint arXiv:2302.09173*, 2023.

[22] Andrea Madotto, Zihan Liu, Zhaojiang Lin, and Pascale Fung. Language models as few-shot learner for task-oriented dialogue systems. *CoRR*, abs/2008.06239, 2020.

[23] Shrikant Malviya, Piyush Kumar, Suyel Namasudra, and Uma Shanker Tiwary. Experience replay-based deep reinforcement learning for dialogue management optimisation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, may 2022. Just Accepted.

[24] Donald Michie, Michael Bain, and J Hayes-Miches. Cognitive models from subcognitive skills. *IEE control engineering series*, 44:71–99, 1990.

[25] Tomáš Nekvinda and Ondřej Dušek. Shades of bleu, flavours of success: The case of multiwoz, 2021.

[26] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. Few-shot natural language generation for task-oriented dialog, 2020.

[29] Dinesh Raghu, Shantanu Agarwal, Sachindra Joshi, et al. End-to-end learning of flowchart grounded task-oriented dialogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4348–4366, 2021.

[30] Abhiramon Rajasekharan, Yankai Zeng, Parth Padalkar, and Gopal Gupta. Reliable natural language understanding with large language models and answer set programming, 2023.

[31] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 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, 2020.

[32] Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. proScript: Partially Ordered Scripts Generation. In *Findings of EMNLP*, 2021.

[33] Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Hierarchical Reinforcement Learning for Zero-shot Generalization with Subtask Dependencies. In *NeurIPS*, 2018.

[34] Sungryull Sohn, Hyunjae Woo, Jongwook Choi, and Honglak Lee. Meta Reinforcement Learning with Autonomous Inference of Subtask Dependencies. In *ICLR*, 2020.

[35] Sungryull Sohn, Hyunjae Woo, Jongwook Choi, Lyubing Qiang, Izzeddin Gur, Aleksandra Faust, and Honglak Lee. Fast Inference and Transfer of Compositional Task Structures for Few-shot Task Generalization. In *UAI*, 2022.

[36] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system. *arXiv preprint arXiv:2006.06814*, 2020.

[37] Weizhi Wang, Zhirui Zhang, Junliang Guo, Yinpei Dai, Boxing Chen, and Weihua Luo. Task-oriented dialogue system as natural language generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2698–2703, New York, NY, USA, 2022. Association for Computing Machinery.

[38] Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. Tod-bert: Pre-trained natural language understanding for task-oriented dialogue. *arXiv preprint arXiv:2004.06871*, 2020.

[39] Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63(10):2011–2027, 2020.

[40] Li Zhou, Kevin Small, Oleg Rokhlenko, and Charles Elkan. End-to-end offline goal-oriented dialog policy learning via policy gradient, 2017.

A Experiment Details

A.1 Next Action Prediction

A.1.1 Dataset Preprocessing details

In the next action prediction experiment, we are using 2 datasets: MultiWOZ [4] and SGD [31]. We first split each dataset into domains and train/test splits. For MultiWOZ, the dataset has an official splitting of train/val/test splits, so we follow the same splits; while MultiWOZ contains dialogs from 7 domains, 2 of them (police, hospital) have no test dialogs, thus we split MultiWOZ dialogs into 14 domains, including 5 single-domains (i.e. dialogs that only involves one of the 5 domains) and 9 multi-domains (i.e. dialogs that involves multiple domains, such as Hotel+Train). For SGD, since the official train/val/test splits often involves test schemas that does not exist in the train set, we decided to create our own train/test splits from the official training set. There are 24 different schemas that have single-schema dialogs in the official training set, so we treat each of these schemas as a separate domain and randomly split dialogs within each domain into train/test splits at a 9:1 ratio.

We then turn each dialog into a trajectory (as defined in section 3.1). Below is how we define the dialog actions within each domain of each dataset:

For SGD, since there are already very comprehensive dialog acts and slot annotations, we directly use the acts defined in the dataset (with a few re-naming) plus the slot annotations to build our set of all possible actions. In addition, since SGD provides explicit annotations about system's database queries, whenever the system queries database, we add an additional turn in our trajectory with one action "SYSTEM query <Intent>" and the status update would be either query success or query failure.

For MultiWOZ, we mostly also directly use the acts and slots in the dataset annotation to build our set of all possible actions, but we did some re-naming and re-organization to remove some redundant combinations of acts and slots (for example, "SYSTEM Booking-Inform <slot>" was changed into "SYSTEM OfferBook + SYSTEM inform <slot>"). Since there are no explicit annotation about database querying, we simply assume that the system looks up information before each "book/nobook" operation and add a corresponding status update to the utterance before these actions. We do not explicitly add "query" actions or additional turns to the trajectories.

Then, within each domain, we will use the trajectories of the train split dialogs to obtain graphs and also act as demonstrations for LLMs, and use the graphs to improve next action predictions on the test dialogs.

A.1.2 Large Language Model Prompting Details

In this experiment, we used GPT-turbo-3.5 as well as FLAN-T5 [7] as baseline next action predictors. We prompt the two LLMs using the exact same prompting method. For each domain, we first randomly select 10 dialogs from the training split, and then use their trajectories (i.e. actions and statuses of each turn) as demonstrations. Then for every system utterance in the test-split dialogs in the domain, we include as many of the demonstration trajectories as possible without exceeding the max token limit of the LLMs, and then we include the partial trajectory of the test dialog up to the turn where the next system actions needs to be predicted. Lastly, we ask the LLM to predict the next system actions, and we programmatically parse the results into individual action items. See Figure 4 for example prompt for SGD and MultiWOZ dataset.

When obtaining the baseline result for each model, we use the top prediction by probability by setting temperature to zero (thus the language model's generation is deterministic). When we need to multiple predictions for graph filtering, for FLAM-T5-xxl we simply do a beam-search of size 10; for GPT-turbo-3.5 we set temperature to 1 and sample 10 times.

A.2 Response Generation

A.2.1 Getting Alternates from each model

For Galaxy [12], we used the top choice of beam-size-1 as well as the top 3 choices of beam-size-5 as alternates. The overall ranking of the 5 choices from highest to lowest are baseline, beam-size-1, beam-size-5 top choice, beam-size-5 second choice, beam-size-5 third choice.

For HDNO [36], since the official prediction baseline is the top choice of beam-size-5, we use the two choices from beam-size-2 and the second and third choice from beam-size-5 as alternates. The overall ranking of the 5 choices from highest to lowest are baseline (i.e. beam-size-5 top choice), beam-size-2 top choice, beam-size-5 second choice, beam-size-2 second choice, beam-size-5 third choice.

For HDSA [6] the process is slightly different. HDSA model consists of 2 parts: the first part (predictor) predicts the actions the system will perform (although in a very different format than what we do in our next action prediction experiment, so not directly comparable), and the second part (generator) uses the output of the predictor to generate the response. If we fix the predictor output and do beam-search on the generator only, the actions within the generated response will almost always be identical, which renders our method useless. Therefore, we created our alternates by tweaking a hyperparameter in the predictor a little bit. The output of the predictor is a binary vector, and the post-sigmoid logits of the predictor is converted to the binary vector by a threshold. The HDSA official code repository has 0.4 as the default threshold, and we changed the threshold around that value and used the different generated vectors as inputs to the generator to obtain our alternates. The overall ranking of the 5 choices from highest to lowest are baseline, threshold-0.4, threshold-0.375, threshold-0.35, threshold-0.325.

A.2.2 Evaluation Details

We use the official MultiWOZ_Evaluation repository to evaluate the BLEU/INFORM/SUCCESS metrics. Since our policy-learning setting assumes that we have access to the ground truth dialog state before the utterance, we use the ground truth dialog state and active domains in the evaluation scripts (by removing dialog state / active domain predictions and only including the response in the prediction file). This is necessary because we found that active domain predictions affect the INFORM/SUCCESS metrics, and incorrect active domain can increase/decrease INFORM/SUCCESS randomly. Therefore, to ensure fairness and consistency, we always use the ground truth active domain during evaluation.

A.2.3 Using GPT as NLU unit

We prompt GPT-turbo-3.5 to convert the candidate responses into action sets. For responses for dialogs in each domain, We first provide randomly selected dialogs from the training split of the same domain together with their ground truth system actions as demonstrations. Then, we specify the desired output format and provide an example of the output format. Lastly, we provide the dialog history of the current candidate responses, and ask GPT to give us actions to all candidate responses (i.e. the baseline + 4 sampled from the models). We start with 6 demonstrations, and we reduce the number of demonstrations by one iteratively if the total number of tokens exceeds the maximum token limit of the model (4097). We show one example prompt for responses from Galaxy [12] together with the GPT completion in Figure 5.

We evaluated the quality of this NLU process by using this process to predict actions of the ground truth responses and compare the predicted actions to the ground truth actions on a subset of the test dialogs. We found that on average our NLU's predicted actions achieves an average F-1 score of 77.6%, which is okay but far from perfect, and the imperfectness of our NLU brings additional challenge to our task.

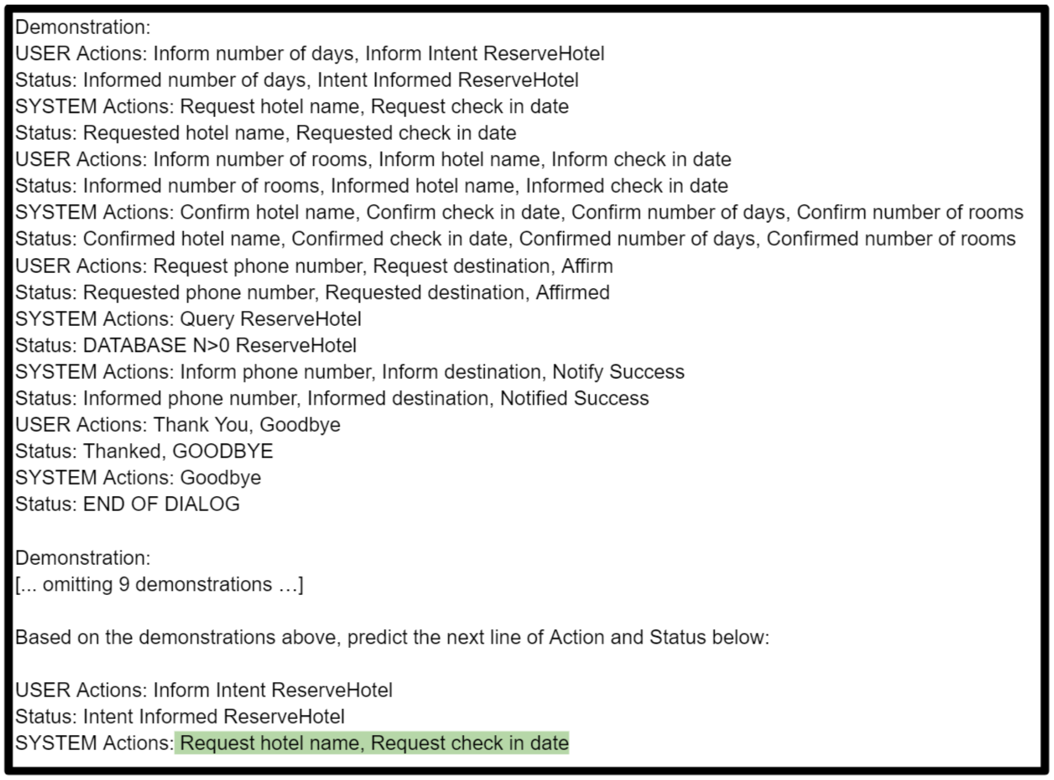
B Human-drawn GT graphs for SGD

In order to perform qualitative assessment of the quality of our graphs, we manually drew graphs for 10 domains in SGD. We show 2 such graphs in Figure 6 for RideSharing_1 schema and Figure 7 for RentalCars_1 schema respectively.

SNG0772 Fourth Turn:
... (USER and SYSTEM talks about a hotel)
USER: No, I'm not ready to book it yet. Thank you!

	Utterance	Actions
Ground Truth	Just let us know when you are ready and we will be happy to assist you. Have a great day!	SYSTEM general-reqmore; SYSTEM offerbook
GALAXY Top/Official Predicted Response	the address is [value_address], postcode [value_postcode]. the phone number is [value_phone]	SYSTEM hotel-inform addr; SYSTEM hotel-inform post; SYSTEM hotel-inform phone
Our chosen Alternate GALAXY Prediction	is there anything else i can help you with?	SYSTEM general-reqmore

Table 5: An example response generation where our graphs helped us select a better response from alternate response predictions. The "inform phone" action from the top/official response is not in "Can" according to our graph (since the user never asked for it) and did not include "reqmore" action which is in "Shd" according to the graph (since you should ask if the user needs anything else after the user ended a particular intent), and these violations caused us to select an alternate that do not violate our graph, and the alternate turns out to be better and closer to the ground truth.

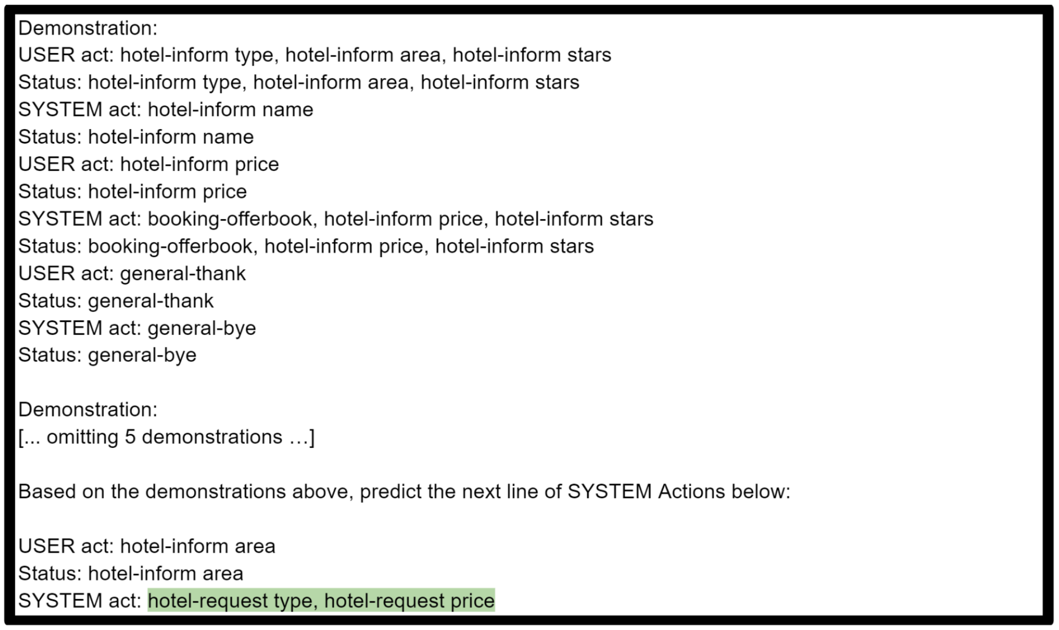


```
Demonstration:
USER Actions: Inform number of days, Inform Intent ReserveHotel
Status: Informed number of days, Intent Informed ReserveHotel
SYSTEM Actions: Request hotel name, Request check in date
Status: Requested hotel name, Requested check in date
USER Actions: Inform number of rooms, Inform hotel name, Inform check in date
Status: Informed number of rooms, Informed hotel name, Informed check in date
SYSTEM Actions: Confirm hotel name, Confirm check in date, Confirm number of days, Confirm number of rooms
Status: Confirmed hotel name, Confirmed check in date, Confirmed number of days, Confirmed number of rooms
USER Actions: Request phone number, Request destination, Affirm
Status: Requested phone number, Requested destination, Affirmed
SYSTEM Actions: Query ReserveHotel
Status: DATABASE N>0 ReserveHotel
SYSTEM Actions: Inform phone number, Inform destination, Notify Success
Status: Informed phone number, Informed destination, Notified Success
USER Actions: Thank You, Goodbye
Status: Thanked, GOODBYE
SYSTEM Actions: Goodbye
Status: END OF DIALOG

Demonstration:
[... omitting 9 demonstrations …]

Based on the demonstrations above, predict the next line of Action and Status below:

USER Actions: Inform Intent ReserveHotel
Status: Intent Informed ReserveHotel
SYSTEM Actions: Request hotel name, Request check in date
```



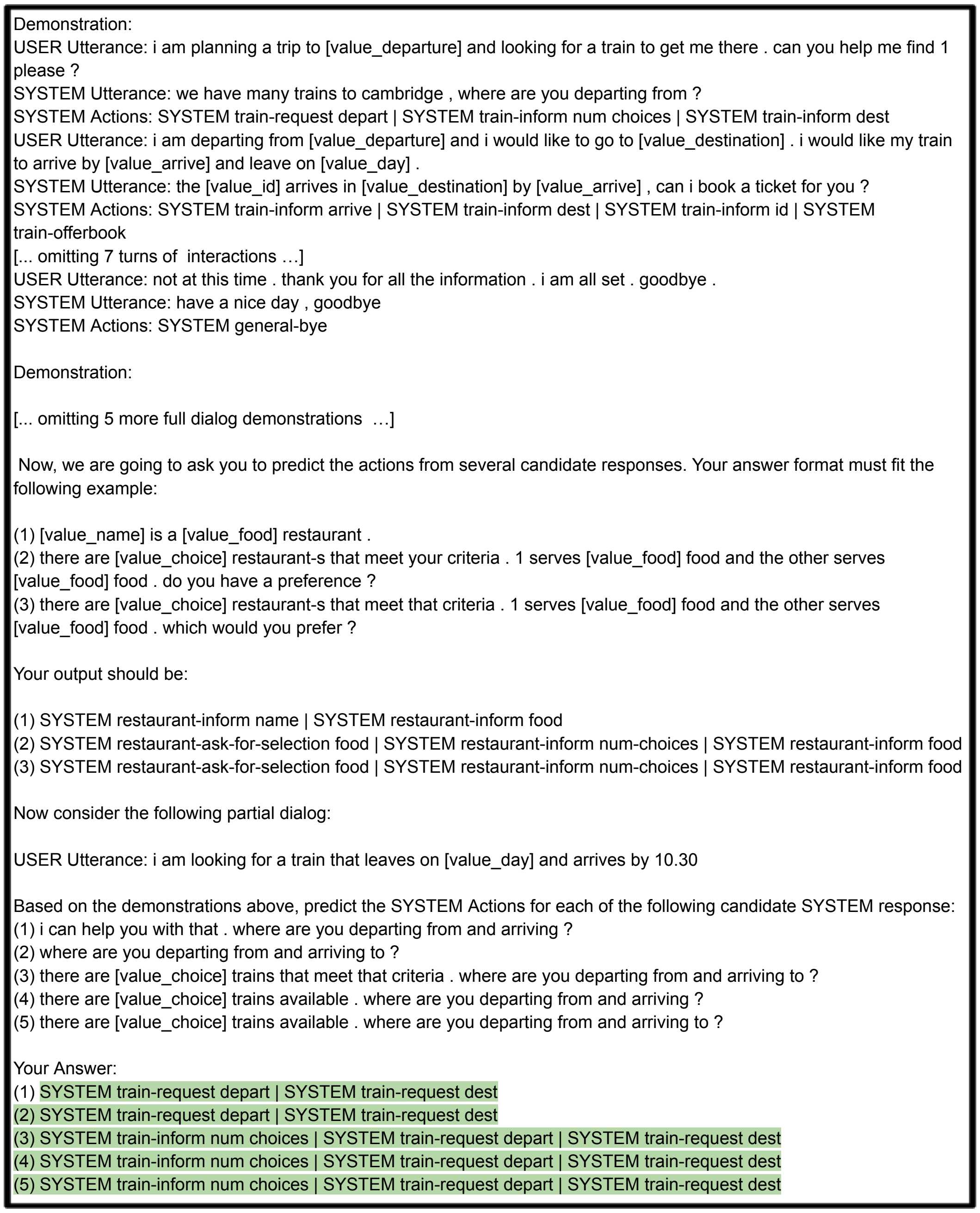
```
Demonstration:
USER act: hotel-inform type, hotel-inform area, hotel-inform stars
Status: hotel-inform type, hotel-inform area, hotel-inform stars
SYSTEM act: hotel-inform name
Status: hotel-inform name
USER act: hotel-inform price
Status: hotel-inform price
SYSTEM act: booking-offerbook, hotel-inform price, hotel-inform stars
Status: booking-offerbook, hotel-inform price, hotel-inform stars
USER act: general-thank
Status: general-thank
SYSTEM act: general-bye
Status: general-bye

Demonstration:
[... omitting 5 demonstrations …]

Based on the demonstrations above, predict the next line of SYSTEM Actions below:

USER act: hotel-inform area
Status: hotel-inform area
SYSTEM act: hotel-request type, hotel-request price
```

Figure 4: Example of using GPT-turbo-3.5 to do next system action prediction from the Hotels_1 domain of SGD (top) and from the Hotel+Train domain of MultiWOZ (bottom). We first provide a few demonstration trajectories in the prompts, then we ask the large language model to complete the prompt. The text highlighted in green is the GPT completion, while everything before that is our input prompt. We give the exact same prompt to the other LLM (FLAN-T5-xxl).



```
Demonstration:
USER Utterance: i am planning a trip to [value_departure] and looking for a train to get me there . can you help me find 1
please ?
SYSTEM Utterance: we have many trains to cambridge , where are you departing from ?
SYSTEM Actions: SYSTEM train-request depart | SYSTEM train-inform num choices | SYSTEM train-inform dest
USER Utterance: i am departing from [value_departure] and i would like to go to [value_destination] . i would like my train
to arrive by [value_arrive] and leave on [value_day] .
SYSTEM Utterance: the [value_id] arrives in [value_destination] by [value_arrive] , can i book a ticket for you ?
SYSTEM Actions: SYSTEM train-inform arrive | SYSTEM train-inform dest | SYSTEM train-inform id | SYSTEM
train-offerbook
[... omitting 7 turns of  interactions …]
USER Utterance: not at this time . thank you for all the information . i am all set . goodbye .
SYSTEM Utterance: have a nice day , goodbye
SYSTEM Actions: SYSTEM general-bye

Demonstration:

[... omitting 5 more full dialog demonstrations  …]

 Now, we are going to ask you to predict the actions from several candidate responses. Your answer format must fit the
following example:

(1) [value_name] is a [value_food] restaurant .
(2) there are [value_choice] restaurant-s that meet your criteria . 1 serves [value_food] food and the other serves
[value_food] food . do you have a preference ?
(3) there are [value_choice] restaurant-s that meet that criteria . 1 serves [value_food] food and the other serves
[value_food] food . which would you prefer ?

Your output should be:

(1) SYSTEM restaurant-inform name | SYSTEM restaurant-inform food
(2) SYSTEM restaurant-ask-for-selection food | SYSTEM restaurant-inform num-choices | SYSTEM restaurant-inform food
(3) SYSTEM restaurant-ask-for-selection food | SYSTEM restaurant-inform num-choices | SYSTEM restaurant-inform food

Now consider the following partial dialog:

USER Utterance: i am looking for a train that leaves on [value_day] and arrives by 10.30

Based on the demonstrations above, predict the SYSTEM Actions for each of the following candidate SYSTEM response:
(1) i can help you with that . where are you departing from and arriving ?
(2) where are you departing from and arriving to ?
(3) there are [value_choice] trains that meet that criteria . where are you departing from and arriving to ?
(4) there are [value_choice] trains available . where are you departing from and arriving ?
(5) there are [value_choice] trains available . where are you departing from and arriving to ?

Your Answer:
(1) SYSTEM train-request depart | SYSTEM train-request dest
(2) SYSTEM train-request depart | SYSTEM train-request dest
(3) SYSTEM train-inform num choices | SYSTEM train-request depart | SYSTEM train-request dest
(4) SYSTEM train-inform num choices | SYSTEM train-request depart | SYSTEM train-request dest
(5) SYSTEM train-inform num choices | SYSTEM train-request depart | SYSTEM train-request dest
```

Figure 5: Example of using GPT-turbo-3.5 to obtain actions from candidate responses from Galaxy [12]. The parts highlighted in green are completion from GPT, and everything before that is our prompt.

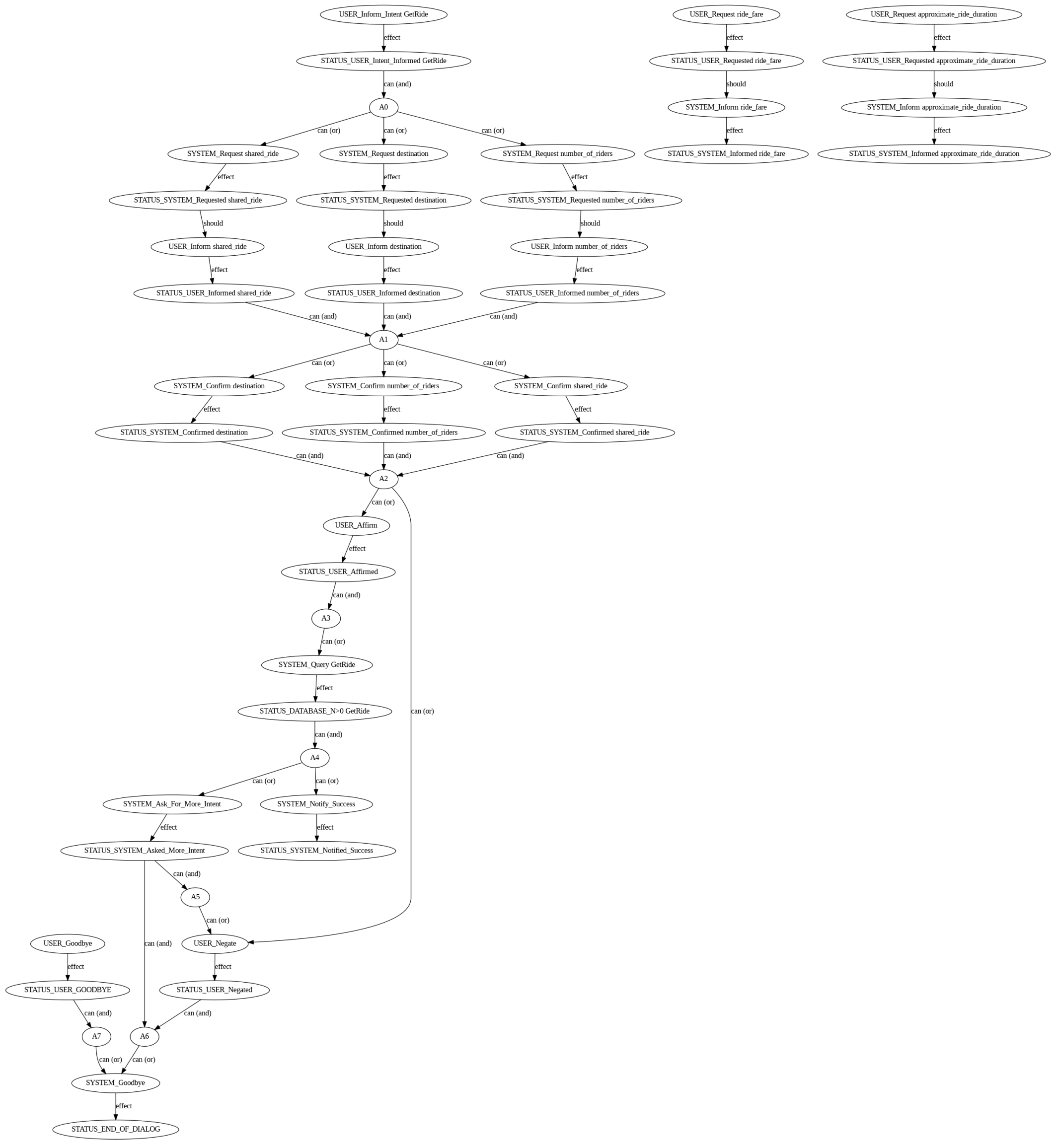

USER_Inform_Intent GetRide
effect
STATUS_USER_Intent_Informed GetRide
can (and)
A0
can (or)
SYSTEM_Request shared_ride
SYSTEM_Request destination
SYSTEM_Request number_of_riders
STATUS_SYSTEM_Requested shared_ride
STATUS_SYSTEM_Requested destination
STATUS_SYSTEM_Requested number_of_riders
should
USER_Inform shared_ride
USER_Inform destination
USER_Inform number_of_riders
STATUS_USER_Informed shared_ride
STATUS_USER_Informed destination
STATUS_USER_Informed number_of_riders
A1
SYSTEM_Confirm destination
SYSTEM_Confirm number_of_riders
SYSTEM_Confirm shared_ride
STATUS_SYSTEM_Confirmed destination
STATUS_SYSTEM_Confirmed number_of_riders
STATUS_SYSTEM_Confirmed shared_ride
A2
USER_Affirm
STATUS_USER_Affirmed
A3
SYSTEM_Query GetRide
STATUS_DATABASE_N>0 GetRide
A4
SYSTEM_Ask_For_More_Intent
SYSTEM_Notify_Success
STATUS_SYSTEM_Asked_More_Intent
STATUS_SYSTEM_Notified_Success
A5
USER_Negate
STATUS_USER_Negated
USER_Goodbye
STATUS_USER_GOODBYE
A7
A6
SYSTEM_Goodbye
STATUS_END_OF_DIALOG
USER_Request ride_fare
STATUS_USER_Requested ride_fare
SYSTEM_Inform ride_fare
STATUS_SYSTEM_Informed ride_fare
USER_Request approximate_ride_duration
STATUS_USER_Requested approximate_ride_duration
SYSTEM_Inform approximate_ride_duration
STATUS_SYSTEM_Informed approximate_ride_duration

Figure 6: Human-drawn ground truth graph for RideSharing_1 domain in SGD.

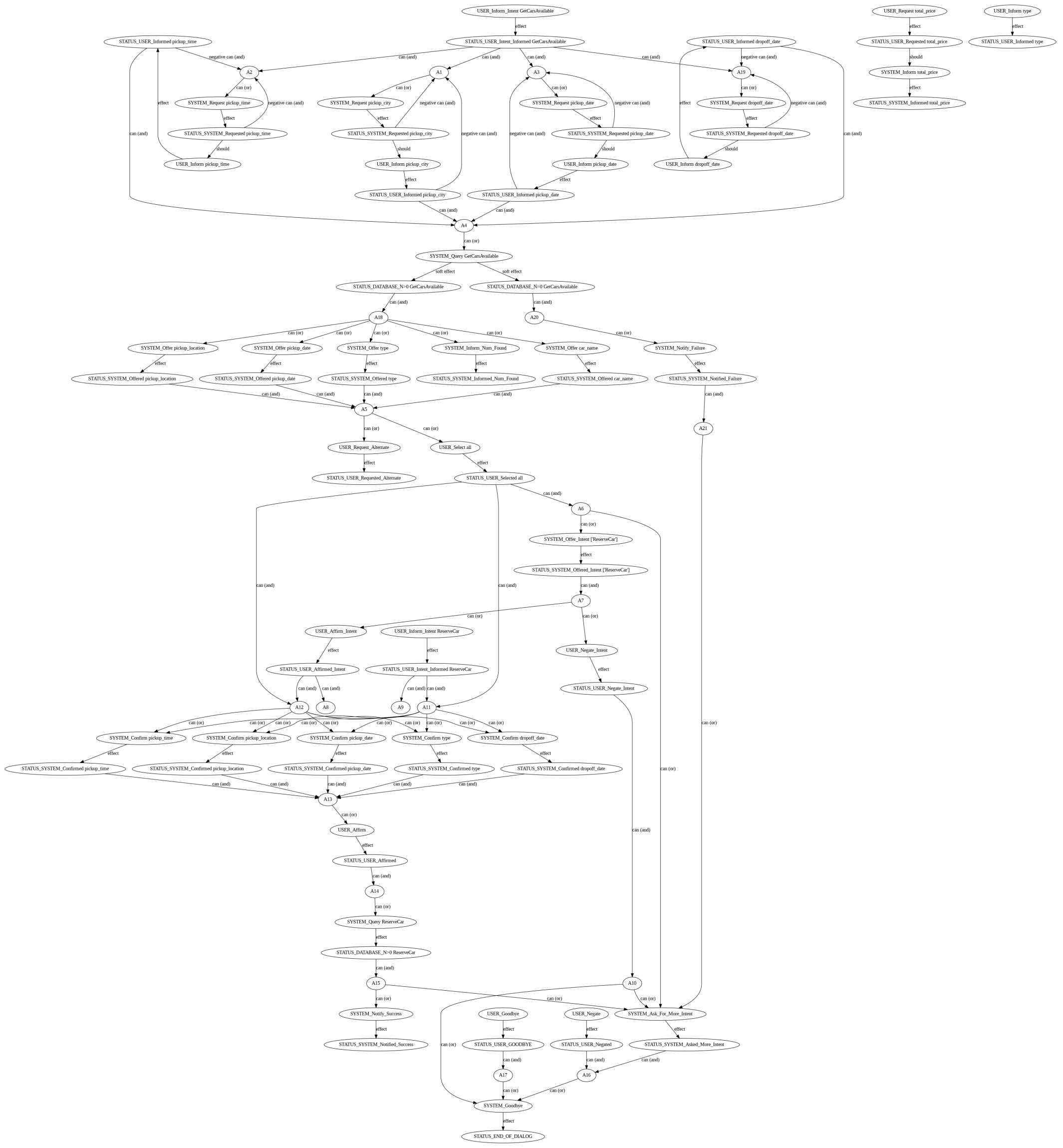

USER_Inform_Intent GetCarsAvailable
STATUS_USER_Intent_Informed GetCarsAvailable
USER_Request total_price
STATUS_USER_Requested total_price
SYSTEM_Inform total_price
STATUS_SYSTEM_Informed total_price
USER_Inform type
STATUS_USER_Informed type
STATUS_USER_Informed pickup_time
SYSTEM_Request pickup_time
STATUS_SYSTEM_Requested pickup_time
USER_Inform pickup_time
SYSTEM_Request pickup_city
STATUS_SYSTEM_Requested pickup_city
USER_Inform pickup_city
STATUS_USER_Informed pickup_city
SYSTEM_Request pickup_date
STATUS_SYSTEM_Requested pickup_date
USER_Inform pickup_date
STATUS_USER_Informed pickup_date
STATUS_USER_Informed dropoff_date
SYSTEM_Request dropoff_date
STATUS_SYSTEM_Requested dropoff_date
USER_Inform dropoff_date
SYSTEM_Query GetCarsAvailable
STATUS_DATABASE_N>0 GetCarsAvailable
STATUS_DATABASE_N=0 GetCarsAvailable
SYSTEM_Offer pickup_location
STATUS_SYSTEM_Offered pickup_location
SYSTEM_Offer pickup_date
STATUS_SYSTEM_Offered pickup_date
SYSTEM_Offer type
STATUS_SYSTEM_Offered type
SYSTEM_Inform_Num_Found
STATUS_SYSTEM_Informed_Num_Found
SYSTEM_Offer car_name
STATUS_SYSTEM_Offered car_name
SYSTEM_Notify_Failure
STATUS_SYSTEM_Notified_Failure
USER_Request_Alternate
STATUS_USER_Requested_Alternate
USER_Select all
STATUS_USER_Selected all
SYSTEM_Offer_Intent ['ReserveCar']
STATUS_SYSTEM_Offered_Intent ['ReserveCar']
USER_Affirm_Intent
STATUS_USER_Affirmed_Intent
USER_Inform_Intent ReserveCar
STATUS_USER_Intent_Informed ReserveCar
USER_Negate_Intent
STATUS_USER_Negate_Intent
SYSTEM_Confirm pickup_time
STATUS_SYSTEM_Confirmed pickup_time
SYSTEM_Confirm pickup_location
STATUS_SYSTEM_Confirmed pickup_location
SYSTEM_Confirm pickup_date
STATUS_SYSTEM_Confirmed pickup_date
SYSTEM_Confirm type
STATUS_SYSTEM_Confirmed type
SYSTEM_Confirm dropoff_date
STATUS_SYSTEM_Confirmed dropoff_date
USER_Affirm
STATUS_USER_Affirmed
SYSTEM_Query ReserveCar
STATUS_DATABASE_N>0 ReserveCar
SYSTEM_Notify_Success
STATUS_SYSTEM_Notified_Success
USER_Goodbye
STATUS_USER_GOODBYE
USER_Negate
STATUS_USER_Negated
SYSTEM_Ask_For_More_Intent
STATUS_SYSTEM_Asked_More_Intent
SYSTEM_Goodbye
STATUS_END_OF_DIALOG

Figure 7: Human-drawn ground truth graph for RentalCars_1 domain in SGD.