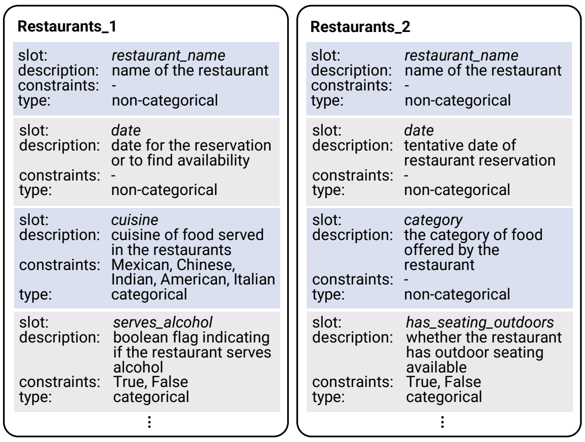
End-to-End Task-Oriented Dialogue Systems Based on Schema

Wiradee Imrattanatrai and **Ken Fukuda**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
`wiradee.imrattanatrai@aist.go.jp and ken.fukuda@aist.go.jp`

Abstract

This paper presents a schema-aware end-to-end neural network model for handling task-oriented dialogues based on a dynamic set of slots within a schema. Contrary to existing studies that proposed end-to-end approaches for task-oriented dialogue systems by relying on a unified schema across domains, we design our approach to support a domain covering multiple services where diverse schemas are available. To enable better generalizability among services and domains with different schemas, we supply the schema's context information including slot descriptions and value constraints to the model. The experimental results on a well-known Schema-Guided Dialogue (SGD) dataset demonstrated the performance improvement by the proposed model compared to state-of-the-art baselines in terms of end-to-end modeling, dialogue state tracking task, and generalization on new services and domains using a limited number of dialogues.[1]



Restaurants_1

slot: *restaurant_name*
description: name of the restaurant
constraints: -
type: non-categorical

slot: *date*
description: date for the reservation or to find availability
constraints: -
type: non-categorical

slot: *cuisine*
description: cuisine of food served in the restaurants
constraints: Mexican, Chinese, Indian, American, Italian
type: categorical

slot: *serves_alcohol*
description: boolean flag indicating if the restaurant serves alcohol
constraints: True, False
type: categorical

⋮

Restaurants_2

slot: *restaurant_name*
description: name of the restaurant
constraints: -
type: non-categorical

slot: *date*
description: tentative date of restaurant reservation
constraints: -
type: non-categorical

slot: *category*
description: the category of food offered by the restaurant
constraints: -
type: non-categorical

slot: *has_seating_outdoors*
description: whether the restaurant has outdoor seating available
constraints: True, False
type: categorical

⋮

Figure 1: A set of slots with their descriptions, value constraints, and types within schemas of the two different services for **Restaurants** domain

1 Introduction

Task-oriented dialogue (TOD) systems enable a flexible alternative for users to fulfill tasks, such as flight booking, arranging transportation, and checking the weather, through a natural language conversation with a virtual assistant. Due to the potential benefits of such a system for both users and industries, an increasing number of application domains and individual services within the domain are available for the system to support. Typically, each service can be distinguished by its schema which determines different functionalities and criteria for the task. To maintain user engagement and satisfaction, it is important for the system to be adjustable to a variety of schemas to effectively interpret the user's goal as well as deliver appropriate responses.

To promote the system's scalability and transfer ability, recent studies proposed approaches that utilize a pre-trained language model (Peng et al., 2021; Yang et al., 2021; Lee, 2021) such as GPT-2 (Radford et al., 2018) and T5 (Raffel et al., 2020). These approaches identify task-specific slots and values within a user input utterance, track belief states (*i.e.* user's goals), retrieve database records using current belief states, estimate an appropriate system action, and translate the system action to a natural language response in an end-to-end manner. These approaches allow the system to be generalized to a new schema of a new domain, however, they are based on the assumption that each domain is associated with a unified schema where its schema elements (*i.e.* slots) are commonly shared across domains. For example, *address* and *postcode* are used as shared slots for both **Hotels** and **Restaurants** domains. This assumption does not necessarily reflect a real-world scenario where multiple services with different schemas exist for a single domain. Each service schema can consist of slots that are semantically similar to slots of other services, and slots that are specific to its own service. As illustrated in Figure 1, while **Restaurants_1** and **Restaurants_2** services of **Restaurants** domain share the same

---

[1] The code is available at `https://github.com/aistairc/SchemaTOD`.

Findings of the Association for Computational Linguistics: ACL 2023, pages 10148–10161
July 9-14, 2023 ©2023 Association for Computational Linguistics

slots such as *restaurant_name* and *date*, there are also slots such as *cuisine* of **Restaurants_1** and *category* of **Restaurants_2** which correspond to a similar interpretation and usage, but their surfaces are different. In addition, **Restaurants_1** has a specific slot, *serves_alcohol*, whereas **Restaurants_2** has *has_seating_outdoors* specified in its schema.

Without addressing the differences in schemas of multiple services of the same domain, the system would unsuccessfully associate values within the user utterance with relevant slots that are applicable for each service. This could lead to inaccurate tracking of the user's goal which in turn, causes the system to generate an irrelevant response. Therefore, in this work, we proposed a schema-aware end-to-end neural network model, initialized with a pre-trained language model T5 (Raffel et al., 2020), to handle task-oriented dialogues of multiple services and domains with diverse schemas. Our proposed model is adaptable to a dynamic set of slots within a schema as we feed slot descriptions and value constraints as the additional context information, along with the dialogue history to the model. For the experiments, the variations of the proposed model using different schema information were evaluated and compared to recent approaches for end-to-end TOD systems using a publicly available dataset, Schema-Guided Dialogue (SGD) dataset (Rastogi et al., 2020). According to the experimental results, the variations of the proposed model outperformed the baselines in all evaluation metrics in terms of the end-to-end modeling and dialogue state tracking (DST) task. In addition, our proposed model demonstrated improved generalizability on new services and domains under zero-shot and few-shot settings.

2 Related Works

2.1 End-to-end approach

Traditionally, the TOD system relies on a pipeline module comprising four subtasks including natural language understanding (NLU) (Yao et al., 2013; Hashemi et al., 2016; Wu et al., 2020), dialogue state tracking (DST) (Mrkšić et al., 2015, 2017; Chen et al., 2020), dialogue policy (POL) (Peng et al., 2017, 2018), and natural language generation (NLG) (Mi et al., 2019; Peng et al., 2020). However, due to an advanced neural approach, it has been proven that building TOD systems based on a single end-to-end neural model achieved promising results and potentially enables efficient optimization (Wen et al., 2017; Lei et al., 2018; Zhang et al., 2020; Ham et al., 2020; Lin et al., 2020; Hosseini-Asl et al., 2020; Wang et al., 2022; Peng et al., 2021; Yang et al., 2021; Lee, 2021). Among the end-to-end approaches, Wen et al. (2017) first proposes a framework using a sequence-to-sequence architecture (Sutskever et al., 2014) without explicitly modeling the user goal. Similarly, Lei et al. (2018) relies on a sequence-to-sequence architecture with an extendable feature for modeling belief spans. Zhang et al. (2020) considers one-to-many conversational property to generate multiple responses. Another line of work leverages transfer learning using a large-scale pre-trained language model (Peng et al., 2021; Yang et al., 2021; Lee, 2021). However, none of these works consider the presence of multiple services within a single domain and only assume that most schema elements are shared among domains. Thus, their approaches cannot recognize the difference in schemas among services and could not be effectively applied when a new schema of a new service does not correlate with known schemas.

2.2 Dialogue state tracking using context

DST is an essential and challenging task for TOD systems, especially when considering tracking slots on different schemas. One effective approach for this challenge is to utilize cross-domain transfer learning which allows the neural models to be applied to new domains (Wu et al., 2019; Kumar et al., 2020; Lin et al., 2021a).

In addition, some recent works propose to incorporate encoding of slot names or descriptions based on schemas to the model and it demonstrated a greater performance in cross-domain DST (Rastogi et al., 2020; Chen et al., 2020; Zhu et al., 2020; Lee et al., 2021; Lin et al., 2021b). Accordingly, we urge that leveraging context information of slots such as descriptions and value constraints in end-to-end modeling could improve the generalizability of the system on new services and domains.

3 Schema-aware End-to-End Model for Task-Oriented Dialogue Systems

In this section, we describe our proposed schema-aware end-to-end model which is composed of five main components as illustrated in Figure 2. Each component is initialized with either a pre-trained T5 encoder or decoder (Raffel et al., 2020). Our model utilizes slot descriptions and value con-

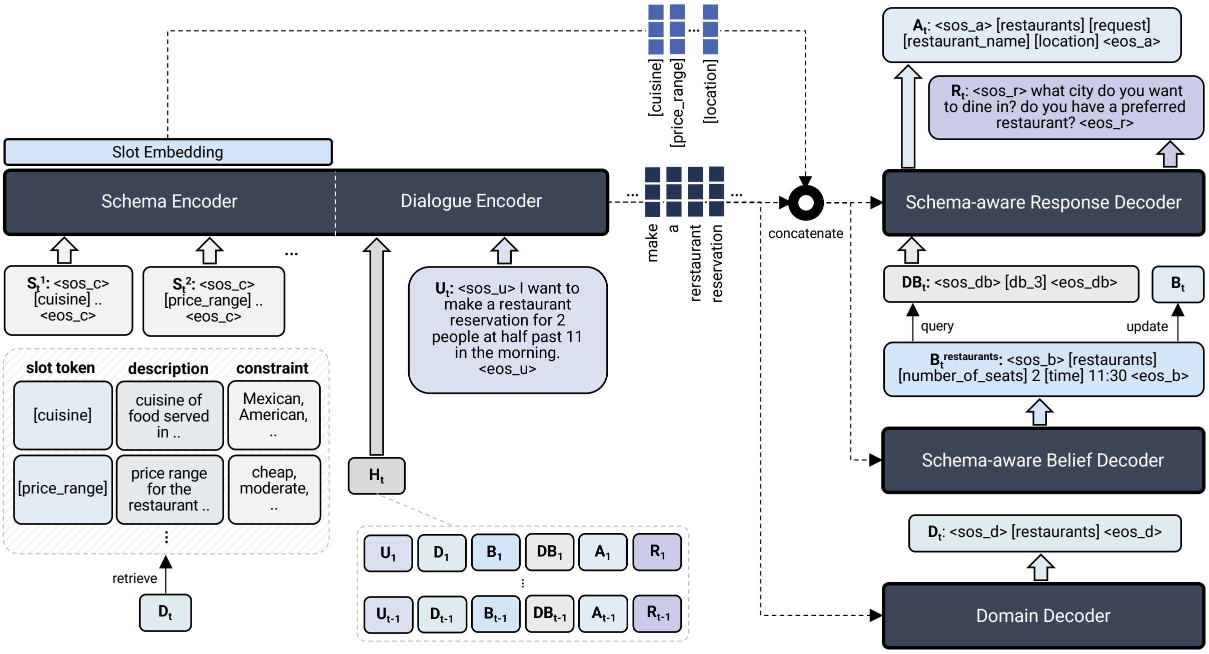

Slot Embedding
Schema Encoder
Dialogue Encoder
[cuisine]
[price_range]
[location]
make
a
restaurant
reservation
concatenate
A_t: <sos_a> [restaurants] [request] [restaurant_name] [location] <eos_a>
R_t: <sos_r> what city do you want to dine in? do you have a preferred restaurant? <eos_r>
Schema-aware Response Decoder
DB_t: <sos_db> [db_3] <eos_db>
B_t
query
update
$B_t^{restaurants}$: <sos_b> [restaurants] [number_of_seats] 2 [time] 11:30 <eos_b>
Schema-aware Belief Decoder
D_t: <sos_d> [restaurants] <eos_d>
Domain Decoder
S_t^1: <sos_c> [cuisine] .. <eos_c>
S_t^2: <sos_c> [price_range] .. <eos_c>
U_t: <sos_u> I want to make a restaurant reservation for 2 people at half past 11 in the morning. <eos_u>
slot token
description
constraint
[cuisine]
cuisine of food served in ..
Mexican, American, ..
[price_range]
price range for the restaurant ..
cheap, moderate, ..
retrieve
D_t
H_t
U_1
D_1
B_1
DB_1
A_1
R_1
U_{t-1}
D_{t-1}
B_{t-1}
DB_{t-1}
A_{t-1}
R_{t-1}

Figure 2: The proposed model architecture composing five main components: (1) a dialogue encoder that encodes an entire dialogue conversation session between a user and system, (2) a schema encoder that encodes slots using their context information within a service schema, (3) a domain decoder that predicts a related domain, (4) a schema-aware belief decoder that decodes a belief span based on slots of a given schema, and (5) a schema-aware response decoder that decodes system action and response

straints to handle a dynamic set of slots within a schema of individual services. The flows to achieve end-to-end modeling based on the five main components are explained in the following subsections:

Dialogue Encoder Initially, the model receives a given dialogue history H_t which includes all dialogue components from turn 1 to $t-1$ and a user utterance U_t at each dialogue turn t to generate an encoded dialogue state through a dialogue encoder.

Different from the previous works which utilize a user utterance U, a belief state B, a database (DB) state DB, a system action A, and a system response R within a dialogue history H, we also include a domain state D as it signals the model with a different focused domain which later used to determine a schema at each turn t. Thus, H_t is constructed by a sequence of U_1, D_1, B_1, DB_1, A_1, $R_1, \ldots, U_{t-1}$, D_{t-1}, B_{t-1}, DB_{t-1}, A_{t-1}, R_{t-1}.

Domain Decoder By using the encoded dialogue state, a domain decoder predicts a domain state D_t where it represents a domain d_t relevant to the given dialogue state. A domain d_t is used as an indicator for retrieving a set of slots S_t based on a service schema.

We opted to predict a domain related to each dialogue turn instead of directly predicting a service because services of the same domain are too similar. This makes it difficult for the model to differentiate. Thus, during inference, we retrieve a set of slots S_t by comparing d_t to the domain of the true service at each turn t. The model will be given a relevant set of slots for the dialogue if a predicted d_t indicates the domain of the true service, otherwise, the model will be given a set of slots belonging to any service of the predicted d_t.

Schema Encoder After obtaining a set of slots S_t within a service schema of a domain, we represent each slot $S_t^i \in S_t$ as a text sequence using its description, value constraint which indicates possible slot values, and slot token. Each sequence is fed to a schema encoder, where its parameters are shared with a dialogue encoder, to generate vector representations. Then, we regard the encoded representation of the last token within a given sequence as an embedding of each slot S_t^i in a schema.

Schema-aware Belief Decoder With the encoded dialogue state and a collection of slot embeddings, the model generates a belief state B_t^d that tracks a user's active goal according to a domain d_t via a schema-aware belief decoder.

As opposed to the existing works where their approaches attempt to generate a belief state representing the user's goals on several domains at once, we focus on tracking changes in the user's goal for a single domain at a time. This is because we observed that the user's goal of a single domain would be updated per each dialogue turn. Therefore, it is nonessential for the decoder to repeat the same user's goal for other inactive domains that do not change. In addition, updating a belief state per domain allows the decoder to pay attention to one set of slots and not mix up with unrelated slots of other services and domains.

B_t^d is then used for querying DB records matching its constraints for constructing a DB state, DB_t, representing a number of matching entities.

At the end of each turn, we update B_t which indicates the user's active goals for multiple related domains for the dialogue by accumulating every B_t^d of every turn up to t.

Schema-aware Response Decoder Lastly, a schema-aware response decoder utilizes the encoded dialogue state by H_t and U_t, a collection of slot embeddings by S_t, as well as a DB state DB_t, to estimate a proper system action A_t and generate a system response R_t in reply to the user utterance U_t.

3.1 Schema-Aware Decoder

In this section, we describe the schema-aware decoder and its importance as the bases for our belief and response decoders in the proposed model.

In general, the decoder would produce a target representation for each target token in a sequence based on attention scores using representations of the input sequence from the encoder, and former target representations by the decoder. Each attention score captures the influence of each token within the input sequence on a target token. Thus, in addition to dialogue token representations $\mathbf{V}_{\text{dial}}$ from a dialogue encoder, we infuse slot embeddings $\mathbf{V}_{\text{slot}}$ based on a service schema as supplementary information to the decoder by $[\mathbf{V}_{\text{slot}}, \mathbf{V}_{\text{dial}}]$, where $[\cdot, \cdot]$ is the concatenation of two matrices. In this way, we expect that the decoder could make use of slot embeddings to estimate the correlation between slots and a target token such that the decoder could infer relevant slots in the generated sequence.

To further encourage predictions among slots of a given service schema, after projecting a final target representation to a fully-connected linear layer, we suppress scores x of target tokens where they indicate the irrelevant slots that do not belong to a given schema. To achieve this, we add weights $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ to scores $x = (x_1, x_2, \ldots, x_n)$, where α_i marks a value zero if i-th presents a relevant slot to a schema, otherwise ∞, and n denotes the number of target tokens.

	Train	Dev	Test
#dialogues	16142	2482	4201
#domains	16	16	18
#services	26	17	21
Avg. #turns per dialogue	20.44	19.63	20.14
Avg. #slots per domain	10.44	8.12	8.5
Avg. #slots per service	8.27	8.0	7.62

Table 1: Statistics of the SGD dataset

3.2 Training Objective

The optimization functions for the domain, belief, and response (system action and response) generation are defined as,

$$\mathcal{L}_{\text{domain}} = -\log p(D_t \mid H_t, U_t) \quad (1)$$

$$\mathcal{L}_{\text{belief}} = -\log p(B_t \mid H_t, U_t, S_t) \quad (2)$$

$$\mathcal{L}_{\text{resp}} = -\log p(A_t, R_t \mid H_t, U_t, S_t, DB_t) \quad (3)$$

Our model is trained to jointly minimize the weighted sum of the three losses, $\mathcal{L} = \mathcal{L}_{\text{domain}} + \mathcal{L}_{\text{belief}} + \mathcal{L}_{\text{resp}}$.

4 Experiments

4.1 Dataset

We conducted the experiments using a well-known dataset, the Schema-Guided Dialogue (SGD) dataset (Rastogi et al., 2020). As opposed to other datasets such as MultiWOZ (Budzianowski et al., 2018) which only defines a single service per domain, the SGD dataset was specifically designed to cover multi-turn dialogue conversations concerning the presence of multiple services per domain. Moreover, as shown in Table 1, the SGD dataset includes a high number of slots per service and domain which is very suitable to our setting as we consider multiple related services per domain and expect the model to be adaptable to various sets of slots within a service schema of a domain.

4.2 Preprocessing

Dialogue components are represented similarly as in the existing works (Yang et al., 2021; Lee, 2021) in which they converted each dialogue information into a span and enclose it with a start tag <sos_$*$> and an end tag <eos_$*$> where $*$ denotes a span's abbreviation. To represent a slot in belief and system action spans, we delexicalized a slot with [slot] placeholder where slot denotes a slot name with an underscore replacing a space since there are slots with long names in the SGD dataset. Additionally, we also replaced a slot value in a response span with [slot] instead of [value_slot] as in the previous works. The reason is that it is not necessary for the model to learn a separate correlation between a slot and a target token and between a slot value and a target token. This helps the model to conveniently determine a slot placeholder for every type of span. For slot context span, we form a text sequence following a template, "<sos_c> t_s [desc] d_s [cons] v_s <eos_c>", where t_s, d_s and v_s denote a slot token, description and value constraint, respectively.

4.3 Experimental Setting

4.3.1 Baselines and Comparisons

We compared our proposed model with the three most recent end-to-end modeling approaches for TOD systems, SOLOIST (Peng et al., 2021), UBAR (Yang et al., 2021) and MTTOD (Lee, 2021). SOLOIST is the model implemented based on GPT-2 with a contrastive objective using negative examples. UBAR is built on a session-level of dialogues based on DistilGPT2 (Sanh et al., 2019), a distilled version of GPT-2. MTTOD relies on a base version of T5 and is separated into three components (*i.e.* a dialogue encoder, belief encoder, and response decoder). We followed the implementation that was released by each baseline to obtain results on the SGD dataset.

For our proposed model, we examine the effectiveness of using different slot context information. The variations include (1) **SchemaTOD (w/o context)** which is the proposed model without exploiting any slot context information, (2) **SchemaTOD (desc)** which is the model using only *slot descriptions*, (3) **SchemaTOD (cons)** which is the model using only *slot value constraints*, and (4) **SchemaTOD (desc, cons)** which is the model where we incorporate both *slot descriptions* and *slot value constraints*.

4.3.2 Evaluation Metrics

To evaluate task completion and the quality of generated responses, we follow the automatic evaluation metrics by Budzianowski et al. (2018) including (1) **BLEU** which measures the fluency of the generated responses (Papineni et al., 2002), (2) **Inform** which measures whether a system has provided accurate entities according to a belief state, (3) **Success** which verifies whether all requested information has been given to the user, and (4) **Combined** as a combined score of the three aforementioned scores to measure an overall quality suggested in Mehri et al. (2019), which is computed as (**Inform** + **Success**) × 0.5 + **BLEU**. In addition, we use **Joint Goal Accuracy (JGA)** to evaluate the performance of the model for the DST task.

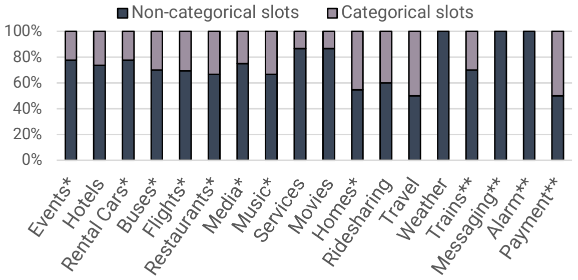


Figure 3: Percentages of non-categorical slots and categorical slots of each domain in the SGD test set. * indicates the domain with no seen services in the training set, while ** indicates the unseen domain.

4.3.3 Implementation Details

We implemented the proposed model based on the base variant of T5. All baselines and model variations were trained using the AdamW (Loshchilov and Hutter, 2018) optimizer with a constant learning rate of 0.0003 for a total of 15 epochs for models operating on a turn-level of dialogues (*i.e.* SOLOIST, MTTOD, and SchemaTOD) and 20 epochs for models operating on a session-level (*i.e.* UBAR). For SOLOIST and UBAR, we trained the models using a batch size of 4 and a gradient accumulation step of 4 on a four-GPUs NVIDIA V100, while we trained MTTOD and variations of SchemaTOD using a batch size of 16 with no gradient accumulation on an eight-GPUs NVIDIA A100. We adopted a simple greedy decoding technique with a temperature of 0.7. For evaluating results on a test set, we choose the best model checkpoint with the best performance based on the **Combined** score on a validation set.

4.4 Experimental Results

4.4.1 End-to-End

Table 2 shows the performance of the baseline approaches and the proposed model using different slot context information for the evaluation of end-to-end modeling. All variations of our Schema-

Model	BLEU	Inform	Success	Combined
SOLOIST	8.32	9.85	6.95	16.72
UBAR	12.60	16.45	11.58	26.61
MTTOD	14.59	15.54	11.00	27.86
SchemaTOD (w/o context)	14.49	25.41	18.98	36.68
SchemaTOD (desc)	14.72	**26.65**	**20.44**	**38.27**
SchemaTOD (cons)	14.51	23.92	17.93	35.44
SchemaTOD (desc, cons)	**15.03**	26.08	19.49	37.81

Table 2: Performances of the baselines and proposed models using different or no context information in terms of BLEU, Inform, Success, and combined score in the end-to-end setting.

Domain	SchemaTOD		
	(desc)	(cons)	(desc, cons)
Events*	17.18	18.39	**19.23**
Hotels	27.76	27.06	**28.78**
Rental Cars*	15.54	14.89	**17.03**
Buses*	13.06	12.54	**13.88**
Flights*	14.88	14.40	**15.40**
Restaurants*	**54.13**	44.00	50.28
Media*	**19.66**	16.95	15.95
Music*	18.56	12.79	**28.15**
Services	**66.01**	56.24	63.70
Movies	**40.83**	36.22	37.59
Homes*	42.60	**43.75**	41.13
Ridesharing	**68.86**	64.34	66.79
Travel	**30.65**	29.41	30.31
Weather	51.47	50.61	**51.48**
Trains**	**11.15**	11.06	11.01
Messaging**	**36.05**	35.03	34.64
Alarm**	**49.33**	48.31	49.25
Payment**	45.13	32.45	**48.77**

Table 3: Performances of the proposed model variations using only slot descriptions, only slot value constraints, and both slot descriptions and value constraints in terms of a combined score by each domain in the test set. * indicates the domain with no seen services, while ** indicates the unseen domain.

TOD outperformed all of the baselines regarding Inform, Success, and combined scores. For BLEU, SchemaTOD (desc) and SchemaTOD (desc, cons) achieved higher BLEU scores compared to BLEU scores achieved by all baselines. This suggests that our proposed SchemaTOD could retrieve more correct entities based on the tracked belief states as well as generate system responses with the requested information more precisely compared to the baselines. As the baselines do not concern with diverse schemas of different services in a domain, they were unable to pick the correct slots that belong to a target service at each dialogue turn. Therefore, the baseline could not successfully retrieve a set of entities and failed to provide the requested information.

Among the variations of SchemaTOD, SchemaTOD (desc, cons) achieved the highest BLEU score at 15.03, while SchemaTOD (desc) achieved the highest Inform, Success, and combined score at 26.65, 20.44, and 38.27, respectively. SchemaTOD (w/o context) without using any slot context information achieved the lowest scores for all evaluation metrics compared to SchemaTOD (desc) and SchemaTOD (desc, cons). Thus, incorporating slot context information allows the proposed model to be more flexible to diverse service schemas which contain different sets of slots for each domain.

To examine more closely how different slot context information contributes to the performance of the TOD system regarding different service schema of the domain, Table 3 presents combined scores achieved by three of our SchemaTOD variations including SchemaTOD (desc), SchemaTOD (cons) and SchemaTOD (desc, cons) for each domain. The results suggest that the highest combined score could be achieved by different variations depending on the domain. While SchemaTOD (desc) which uses only slot descriptions achieved better performance for most domains with seen services during training (four out of six domains with seen services including **Services**, **Movies**, **Ridesharing** and **Travel**), SchemaTOD (desc, cons) using both slot descriptions and value constraints achieved better performance for most domains with only unseen services (five out of eight domains including **Events**, **Rental Cars**, **Buses**, **Flights** and **Music**). For **Homes** domain, the highest combined score could be achieved by SchemaTOD (cons) which uses only slot value constraints. According to Figure 3 which illustrates the portion of categorical

Model	JGA
SOLOIST	14.21
UBAR	23.78
MTTOD	24.93
SchemaTOD (w/o context)	30.56
SchemaTOD (desc)	30.93
SchemaTOD (cons)	30.22
SchemaTOD (desc, cons)	**31.14**

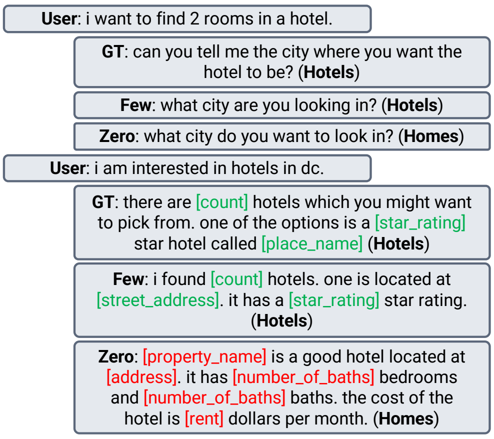
Table 4: Performances of the baselines, proposed models, and variations of proposed models in terms of JGA (%) for evaluation on Dialogue State Tracking task.

and non-categorical slots of each domain in the SGD test set, **Homes** domain has the highest percentage of categorical slots (45.45%) when compared to other domains where only unseen services are available in the test set.

In addition, although SchemaTOD (desc) outperformed other variations of SchemaTOD for most unseen domains (three out of four unseen domains including **Trains**, **Messaging** and **Alarm**), most slots of these domains are non-categorical. For the unseen domain with more categorical slots like **Payment**, SchemaTOD (desc, cons) achieved better performance with the highest combined score. This suggests that using both slot description and value constraints works well for domains with unseen services and unseen domains where the number of categorical slots is significantly high, otherwise using only slot descriptions would suffice. Using slot value constraints would enhance the performance in the case that the schema contains enough useful clues for the possible values of the categorical slots.

4.4.2 Dialogue State Tracking

Apart from the evaluation of the proposed model for the end-to-end modeling, we also evaluate the performance of variations of our SchemaTOD for the DST task which estimates the user's goals based on the dialogues. Table 4 shows the overall performance of baselines and variations of our SchemaTOD in terms of JGA on all test dialogues. SchemaTOD (desc, cons) achieved the highest JGA at 31.14% which is approximately 6.21% higher than the best JGA achieved by the baseline, MTTOD, at 24.93%. The results achieved by the variations of SchemaTOD are comparable to each other with a small difference in JGA.



User: i want to find 2 rooms in a hotel.

GT: can you tell me the city where you want the hotel to be? **(Hotels)**

Few: what city are you looking in? **(Hotels)**

Zero: what city do you want to look in? **(Homes)**

User: i am interested in hotels in dc.

GT: there are [count] hotels which you might want to pick from. one of the options is a [star_rating] star hotel called [place_name] **(Hotels)**

Few: i found [count] hotels. one is located at [street_address]. it has a [star_rating] star rating. **(Hotels)**

Zero: [property_name] is a good hotel located at [address]. it has [number_of_baths] bedrooms and [number_of_baths] baths. the cost of the hotel is [rent] dollars per month. **(Homes)**

Figure 4: The user utterances (User), ground truth responses (GT), and responses generated by the proposed model trained on the few-shot setting (Few) and on the zero-shot setting (Zero) for **Hotels**. The domain in the parentheses at the end of each response indicates the predicted domain for the dialogues at each turn. The slots in green and red are in-schema and out-of-schema slots based on the service for this dialogue.

4.4.3 Generalization on New Services and Domains

To evaluate the model's generalizability on new services and domains with zero or few training examples, we tested the baselines and our proposed SchemaTOD on zero-shot and few-shot settings. Since there are many domains available in the SGD test set, we chose to focus on the performance of the baselines and the best variation of SchemaTOD based on the performance achieved in Table 3 on the top four domains with the highest number of dialogues with unseen services. The top four domains include **Events**, **Hotels**, **Rental Cars**, and **Buses**. The best variation of SchemaTOD is SchemaTOD (desc, cons) for all four domains.

For a zero-shot setting, we held out all dialogues in the training and development sets that involve each target domain and trained all models using the remaining dialogues by following the same procedure as described in Section 4.3.3. Then, we tested the best-trained models on the dialogues involving the target domain within the test set.

For a few-shot setting, we finetuned each model that was trained without exposing it to each target domain by using a random sample of 64 and 32 dialogues of each target domain from the training and development sets, respectively. We finetuned the models for a total of 5 epochs and chose the best model according to the combined score achieved on the random sample dialogues from the development

Model	Events*			Hotels			Rental Cars*			Buses*		
	Zero	Few	All	Zero	Few	All	Zero	Few	All	Zero	Few	All
SOLOIST	5.17	5.50	7.12	6.45	5.49	16.29	4.62	3.79	7.06	4.82	4.25	7.96
UBAR	7.48	9.43	12.02	8.56	8.15	22.66	10.94	10.76	13.35	7.99	7.35	12.64
MTTOD	8.28	10.57	13.68	**10.94**	11.24	23.89	11.16	11.61	14.67	**9.59**	**11.82**	13.87
SchemaTOD	**8.96**	**14.05**	**19.23**	10.10	**15.27**	**28.78**	**12.60**	**16.20**	**17.03**	9.55	11.42	**13.88**

Table 5: Comparison of performances based on a combined score between models trained on the original training set (All), models trained on the training set without a target domain for zero-shot setting (Zero), and models fine-tuned with the target domain for few-shot setting (Few). * indicates the domain with no seen services in the training set.

set. Like the zero-shot setting, we tested the best-fine-tuned models on all dialogues involving the target domain within the test set.

According to Table 5 which demonstrates the performance of the models on different training settings for four target domains, the proposed model, SchemaTOD, achieved the highest combined scores in every domain when the model was trained using the original training set. The highest combined score when using the original test dialogues could be achieved for **Hotels** domain as the test dialogues for this domain involve seen services during training. While under zero-shot and few-shot settings, the combined scores achieved by every model were reduced compared to those achieved by the models trained on the original training set, and the highest scores were achieved mostly by the proposed model. For the zero-shot setting, SchemaTOD performed the best on **Events** and **Rental Cars**, while achieving the highest combined scores on the three domains including **Events**, **Hotels**, and **Rental Cars** for the few-shot setting with the improvement of 4% from the best baseline.

Since our proposed model relies considerably on a retrieved set of slots based on a predicted domain, this makes it challenging for our model to interpret the context of the dialogues when the predicted domain is incorrect and causes the retrieved set of slots to be irrelevant to the dialogues. As suggested by the results for the zero-shot setting, SchemaTOD could perform better for **Events** and **Rental Cars** since these two domains have more slots that are shared with other domains such as *date*, *time*, and *city*. Thus, even though the proposed model failed to utilize the relevant slots, it could still make use of most slots available for other domains interchangeably. On the other hand, the **Hotels** and **Buses** domains have more domain-specific slots such as *check_in_date*, *has_laundry_service*, *has_wifi*, *transfers*, and *additional_luggage*. In this case, the model could not leverage slots of any other domains when the model fails to predict a correct domain for the dialogues.

However, once the model has been exposed to a few dialogue examples of a target domain, SchemaTOD could achieve better performance for most domains in the few-shot setting due to its ability to utilize a relevant set of slots based on the correct predicted domain. Figure 4 presents the actual responses generated by our model under zero-shot and few-shot settings in comparison with the ground truth responses. The second response by the proposed model under the zero-shot setting contains slots that do not belong to the **Hotels** domain since the model predicted **Homes** for the scope of this dialogue. Therefore, this response implies more detail on the available home. In contrast, the second response by the model under the few-shot setting suggests the hotel recommendation which corresponds to the ground truth as the model determines the dialogue as for **Hotels** domain.

5 Conclusion

In this paper, we focus on improving the end-to-end TOD systems to support a domain covering multiple services as well as to achieve better generalizability. We propose SchemaTOD, a schema-aware end-to-end neural network model, which incorporates context information of slots within a service schema of a domain. Experiments conducted on the SGD dataset suggest that our proposed model when using both slot descriptions and value constraints achieved the best performance when the domains consist of only unseen services and when the domains are unseen with more categorical slots. Otherwise, the proposed model using only slot descriptions outperformed other models for the domains with seen services and the unseen domains with a high number of non-categorical slots. For generalizability, the performance achieved by the proposed model demonstrated improvement over the baselines when using a few dialogue examples of the new services and domains.

Limitations

As described in Section 4.4.3, the proposed model requires precise prediction on the target domain at each dialogue turn to utilize the relevant slot information within the service schema. This possesses a difficulty to apply the proposed model to some new domains, especially when the domains share high similarity with seen domains. To reduce this difficulty, as a future improvement, one potential approach is to process on a hierarchical structure of slots within schemas, instead of individual slots. In this way, the model does not need to rely on the domain, but only on a group of similar slots. The model can then perform matching of a chosen group of slots with available slots within a schema and composes responses based on those slots.

Acknowledgements

We thank all anonymous reviewers for their insightful feedback. This work is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7521–7528.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592.

Homa B Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. In *International Conference on Web Search and Data Mining, Workshop on Query Understanding*.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Adarsh Kumar, Peter Ku, Anuj Goyal, Angeliki Metallinou, and Dilek Hakkani-Tur. 2020. Ma-dst: Multi-attention-based scalable dialog state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8107–8114.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949.

Yohan Lee. 2021. Improving end-to-end task-oriented dialog system with a simple auxiliary task. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1296–1303.

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

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Zhenpeng Zhou, Paul Crook, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, and Pascale Fung. 2021a. Zero-shot dialogue state tracking via cross-task transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7890–7900.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021b. Leveraging slot descriptions for zero-shot cross-domain dialogue StateTracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648.

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

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. Structured fusion networks for dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 165–177.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3151–3157.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 794–799.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. 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.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*.

Weizhi Wang, Zhirui Zhang, Junliang Guo, Yinpei Dai, Boxing Chen, and Weihua Luo. 2022. Task-oriented dialogue system as natural language generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2698–2703.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14230–14238.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. 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.

Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020. Efficient context and schema fusion networks for multi-domain dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 766–781.

A Example Outputs

User: help me find somewhere to eat.	**Belief**: - **Response**: sure, where and what kind of food?	GT
	Belief: - **Response**: what city would you like to look in?	MTTOD
	Belief: - **Response**: in which city?	SchemaTOD
User: find an italian restaurant in belmont.	**Belief**: [restaurants] [category] italian [location] belmont **Response**: there's [count] you may like. how about [restaurant_name] in [location]?	GT
	Belief: [restaurants] [city] belmont [cuisine] italian **Response**: i've found [value_count] restaurants in [value_city]. [value_restaurant_name] is a nice one.	MTTOD
	Belief: [restaurants] [location] belmont [category] italian **Response**: there are [count] restaurants in [location]. [restaurant_name] is a nice one.	SchemaTOD
User: what's the phone number?	**Belief**: [restaurants] [category] italian [location] belmont **Response**: they're at [phone_number].	GT
	Belief: [restaurants] [city] belmont [cuisine] italian **Response**: the phone number is [value_phone_number].	MTTOD
	Belief: [restaurants] [location] belmont [category] italian **Response**: it's [phone_number].	SchemaTOD
User: great, thank you.	**Belief**: [restaurants] [category] italian [location] belmont [restaurant_name] caprinos **Response**: should i reserve a table?	GT
	Belief: [restaurants] [restaurant_name] caprinos [city] belmont [cuisine] italian **Response**: would you like to make a reservation?	MTTOD
	Belief: [restaurants] [location] belmont [category] italian [restaurant_name] caprinos **Response**: do you want to reserve a table?	SchemaTOD
User: yeah, reserve a table for 2 for the 14th of this month at morning 11:30.	**Belief**: [restaurants] [category] italian [location] belmont [restaurant_name] caprinos [date] 2019-03-14 [number_of_seats] 2 [time] 11:30 **Response**: sure, please confirm you'd like a table for [number_of_seats] at [restaurant_name] in [location] at [time] on [date].	GT
	Belief: [restaurants] [restaurant_name] caprinos [date] 2019-03-14 [time] 11:30 [party_size] 2 [city] belmont [cuisine] italian **Response**: please confirm that you'd like to book a table for [value_party_size] at [value_restaurant_name] in [value_city] for [value_date] at [value_time].	MTTOD
	Belief: [restaurants] [location] belmont [category] italian [restaurant_name] caprinos [date] 2019-03-14 [number_of_seats] 2 [time] 11:30 **Response**: please confirm: reservation for [number_of_seats] at [restaurant_name] in [location] on [date] at [time].	SchemaTOD
User: correct, is it cheap and what's their address?	**Belief**: [restaurants] [category] italian [location] belmont [restaurant_name] caprinos [date] 2019-03-14 [number_of_seats] 2 [time] 11:30 **Response**: i've reserved your table. they are [price_range] and at [address].	GT
	Belief: [restaurants] [restaurant_name] caprinos [date] 2019-03-14 [time] 11:30 [party_size] 2 [city] belmont [cuisine] italian **Response**: sorry, i was unable to make a reservation. is there anything else i can help with?	MTTOD
	Belief: [restaurants] [location] belmont [category] italian [restaurant_name] caprinos [date] 2019-03-14 [number_of_seats] 2 [time] 11:30 **Response**: reservation complete. it's [price_range] price-wise and located at [address].	SchemaTOD

Table 6: The actual outputs of belief states and responses by MTTOD and SchemaTOD for the first six turns of the dialogue for **Restaurants** domain. The slots in red are out-of-schema slots for **Restaurants_2** service of this domain.

B Domain Accuracy

Table 7 shows accuracy by the four variations of SchemaTOD in predicting domains for the dialogue turns, while Table 8 presents the accuracy by SchemaTOD $(\text{desc}, \text{cons})$ trained on the original training set, and under zero-shot and few-shot settings on the four target domains.

Model	Accuracy
SchemaTOD (w/o context)	79.66
SchemaTOD (desc)	80.73
SchemaTOD (cons)	80.20
SchemaTOD (desc, cons)	81.09

Table 7: Performances of the proposed models using different or no context information on predicting domains in terms of accuracy in percentage.

Domain	SchemaTOD		
	Zero	Few	All
Events*	0	35.11	94.00
Hotels	0	69.72	98.80
Rental Cars*	0	72.91	92.03
Buses*	0	74.28	98.39

Table 8: Domain accuracy between SchemaTOD trained on the original training set (All), the model trained on the training set without a target domain for zero-shot setting (Zero), and the model fine-tuned with the target domain for few-shot setting (Few). * indicates the domain with no seen services in the training set.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section "Limitations".

☒ A2. Did you discuss any potential risks of your work?
We did not collect any data to construct a new dataset. We only use a well-known dataset for task-oriented dialogues which was well-designed and reliable. Thus, we do not involve any human subjects and annotations on our own. All the experiments are done using automatic evaluations. It is unlikely that our work would risk personal information exposure and misconduct.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section "Abstract" and 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4.

☑ B1. Did you cite the creators of artifacts you used?
Section 4.1 and 4.3.1.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The license and terms of use for our code can be found in the supplementary material. Also, for other existing artifacts that we used, the license and terms of use can be found in the original repository addressed in each original paper.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We include an Acknowledgement section in the supplementary material that acknowledges the released code and data of each existing artifact.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We do not discuss the steps of the data collection process as we utilize a publicly available dataset. A detailed explanation of the dataset can be found in the original paper.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4.1.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4.1.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 4.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.3.3.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4.3.3.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4.4.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.3.1, 4.3.2 and 4.3.3.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.