Schema Encoding for Transferable Dialogue State Tracking

Hyunmin Jeon
Computer Science and Engineering
POSTECH, Pohang, South Korea
jhm9507@postech.ac.kr

Gary Geunbae Lee
Computer Science and Engineering
Graduate School of Artificial Intelligence
POSTECH, Pohang, South Korea
gblee@postech.ac.kr

Abstract

Dialogue state tracking (DST) is an essential sub-task for task-oriented dialogue systems. Recent work has focused on deep neural models for DST. However, the neural models require a large dataset for training. Furthermore, applying them to another domain needs a new dataset because the neural models are generally trained to imitate the given dataset. In this paper, we propose **S**chema **E**ncoding for **T**ransferable **D**ialogue **S**tate **T**racking (SET-DST), which is a neural DST method for effective transfer to new domains. Transferable DST could assist developments of dialogue systems even with few dataset on target domains. We use a schema encoder not just to imitate the dataset but to comprehend the schema of the dataset. We aim to transfer the model to new domains by encoding new schemas and using them for DST on multi-domain settings. As a result, SET-DST improved the joint accuracy by 1.46 points on MultiWOZ 2.1.

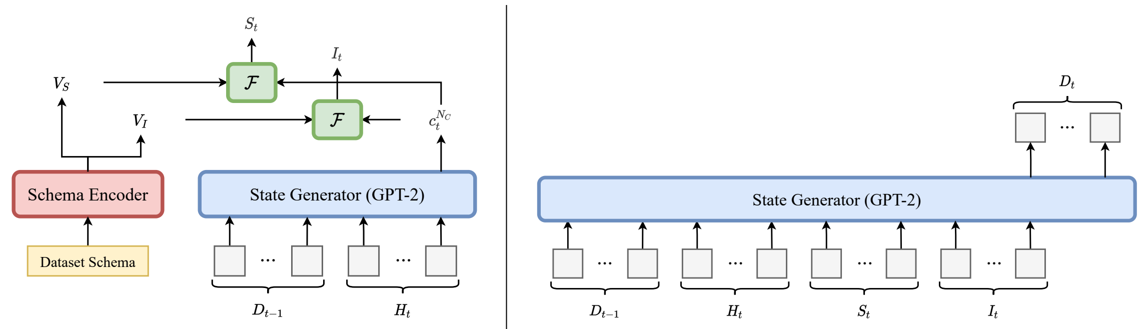
1 Introduction

The objective of task-oriented dialogue systems is to help users achieve their goals by conversations. Dialogue state tracking (DST) is the essential sub-task for the systems to perform the purpose. Users may deliver the details of their goals to the systems during the conversations, e.g., what kind of food they want the restaurant to serve and at what price level they want to book the hotel. Thus, the systems should exactly catch the details from utterances. They should also communicate with other systems by using APIs to achieve users' goals, e.g., to search restaurants and to reserve hotels. The goal of DST is not only to classify the users' intents but also to fill the details into predefined templates that are used to call APIs.

Recent work has used deep neural networks for DST with supervised learning. They have improved the accuracy of DST; however, they require a large dataset for training. Furthermore, they need a new dataset to be trained on another domain. Unfortunately, the large dataset for training a DST model is not easy to be developed in real world. The motivation of supervised learning is to make deep neural networks imitate humans. But, they actually imitate the given datasets rather than humans. Someones who have performed hotel reservation work could easily perform restaurant reservation work if some guidelines are provided, but neural models may have to be trained on a new dataset of the restaurant domain. The difference between humans and neural models is that humans can learn how to read guidelines and to apply the guidelines to their work. This is why transfer learning is important to train neural models on new domains.

In this paper, we propose **S**chema **E**ncoding for **T**ransferable **D**ialogue **S**tate **T**racking (SET-DST), which is a neural DST method with transfer learning by using dataset schemas as guidelines for DST. The motivation of this study is that humans can learn not only how to do their work, but also how to apply the guidelines to the work. We aim to make a neural model learn how to apply the schema guidelines to DST beyond how to fill predefined slots by simply imitating the dataset on multi-domain settings. The schema includes metadata of the dataset, e.g., which domains the dataset covers and which slots have to be filled to achieve goals. SET-DST has a schema encoder to represent the dataset schema, and it uses the schema representation to understand utterances and to fill slots. Recently, transfer learning has been becoming important because development of new datasets is costly. Transfer learning makes it possible to pre-train neural models on large-scale datasets to effectively fine-tune the models on small-scale downstream tasks.

We used SGD (Rastogi et al., 2020) as the large-scale dataset, and evaluated SET-DST on MultiWOZ 2.1 (Eric et al., 2020), which is a standard benchmark dataset for DST, as the downstream task. SET-DST achieved state-of-the-art accuracy



(a) Schema encoding for active slots and intents classification.

(b) Dialogue state generation.

Figure 1: Overview of SET-DST. The schema encoder takes the dataset schema and generates slot vectors and intent vectors. The state generator takes the previous dialogue state D_{t-1} and the dialogue history H_t to calculate active scores of slots and intents. $\mathcal{F}$ is an score function to calculate whether the slots or intents are activated on turn t. Then, the state generator additionally takes the activated slots and intents to generate the current dialogue state D_t. S_t indicates the activated slots and I_t indicates the activated intents.

on the downstream DST task. We further confirmed that SET-DST worked well on the small downstream dataset. This result demonstrates that transfer learning with schema encoding improves the performance of neural DST models and the efficiency of few-shot learning on DST.

2 Related Work

Traditional DST models extract semantics by using natural language understanding (NLU) modules to generate dialogue states (Williams, 2014; Wang and Lemon, 2013). The limitation of these models is that they rely on features extracted by humans.

Recent work has focused on building end-to-end DST models without hand-crafted features. Zhong et al. (2018) use global modules to share parameters between different slots. Nouri and Hosseini-Asl (2018) improve the latency by removing inefficient recurrent layers. Transferable DST models that can be adapted to new domains by removing the dependency on the domain ontology are proposed (Ren et al., 2018; Wu et al., 2019). Zhou and Small (2019) attempt to solve DST as a question answering task using knowledge graph.

More recently, large-scale pre-trained language models such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) are used for DST. The pre-trained BERT acts as an NLU module to understand utterances (Lee et al., 2019; Zhang et al., 2020a; Kim et al., 2020; Heck et al., 2020). GPT-2 makes it possible to solve DST as a conditional language modeling task (Hosseini-Asl et al., 2020; Peng et al., 2021).

Rastogi et al. (2020) propose the baseline method that defines the schema of dataset and uses it for training and inference. A drawback of them is that the calculation cost is high because they use the domain ontology and access all values to estimate the dialogue state. DST models that uses schema graphs to encode the relation between slots and values are proposed (Chen et al., 2020; Zhu et al., 2020). However, they focus on encoding the relation between slots and values of the given domains not on adaptation to new domains.

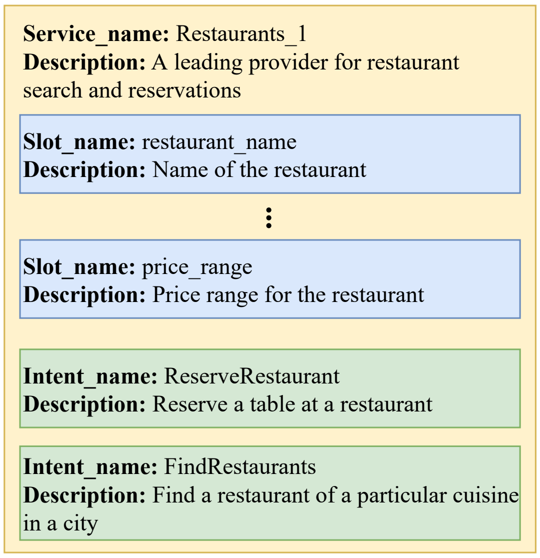
In this paper, we focus on making the model learn how to understand the schema and how to apply it to estimate the dialogue state, not just on encoding the in-domain relation.

3 Schema Encoding for Transferable Dialogue State Tracking

In this section, we describe the architecture of SET-DST and how to optimize it. Figure 1 shows the overview of our method. The model consists of the schema encoder and the state generator. SET-DST generates the dialogue state in two steps: (a) schema encoding and classification, and (b) dialogue state generation. In this paper, we define some terms as follows.

Schema Metadata of the dataset, e.g., what domains, services, slots, and intents the dataset covers. A dataset has a schema that describes the dataset.

Domain What domains the conversation goes on, e.g., restaurant, hotel, and attraction. A conversation can go on multiple domains.



Service_name: Restaurants_1
Description: A leading provider for restaurant search and reservations

Slot_name: restaurant_name
Description: Name of the restaurant

⋮

Slot_name: price_range
Description: Price range for the restaurant

Intent_name: ReserveRestaurant
Description: Reserve a table at a restaurant

Intent_name: FindRestaurants
Description: Find a restaurant of a particular cuisine in a city

Figure 2: Example of schema for restaurant search and reservation service including slots and intents.

Service What services the system provides to users. It is similar to domain, but application-level. For example, restaurant domain can have two different services: (1) a service for searching and reserving restaurants and (2) a service focused on searching and comparing restaurants. In real world, a service corresponds to an application.

Action Abstract actions of users to achieve their goals during conversations, e.g., to inform the system their requirements or to request the system for some information. Appendix B demonstrates the details of the user actions covered in this paper.

Slot The details of the user goals, e.g., the type of food and the price range of hotel. Slots are predefined based on the domains or services that the system should cover, and the slots are filled by DST. The schema includes the information of slots.

Value The values that have actual meaning for the corresponding slots, e.g., cheap or expensive about the price range of hotel. The systems should match slot-value pairs from conversations.

Intent Sub-goals to achieve the final goals of users. A goal consists of one or more intents, and an intent is achieved over one or more conversation turns. In real world, an intent corresponds to an API. For example, to search restaurants or to book hotels should be performed by APIs of external systems. Furthermore, The dialogue system should predict the slot-value pairs which correspond to arguments to call APIs.

3.1 Schema Encoding

We use the pre-trained BERT[1] for the schema encoder. Figure 2 shows an example of the schema for *Restaurant_1* service that is a service to search and reserve restaurants. Services, slots, and intents consist of name and short description. The name and description of the service in the schema are fed into BERT to generate service vector v_R as

$$\begin{aligned} &o_R = \texttt{BERT}\left(\texttt{[CLS]}\, n_R : d_R\, \texttt{[SEP]}\right) \\ &v_R = W_R \cdot o_R^{\texttt{[CLS]}} \in \mathbb{R}^h \end{aligned}, \quad (1)$$

where n_R is the service name, d_R is the service description, and h is the hidden size. $o_R^{\texttt{[CLS]}}$ is the output of `[CLS]` token, and $W_R \in \mathbb{R}^{h \times h}$ is a fully connected (FC) layer. `[CLS]` and `[SEP]` are special tokens that mean the start and end of the sentence, respectively. The service in Figure 2 can be represented as `[CLS] Restaurants_1 : A leading provider for restaurant search and reservations [SEP]` to be fed into BERT. The slots and intents in the schema are also fed into BERT to generate slot vectors $V_S = \{v_S^1, \cdots v_S^{N_S}\} \in \mathbb{R}^{N_S \times h}$ and intent vectors $V_I = \{v_I^1, \cdots, v_I^{N_I}\} \in \mathbb{R}^{N_I \times h}$, respectively, as follows:

$$\begin{aligned} &o_S^j = \texttt{BERT}\left(\texttt{[CLS]}\, n_S^j : d_S^j\, \texttt{[SEP]}\right) \\ &v_S^j = W_S \cdot o_S^{j,\texttt{[CLS]}} \in \mathbb{R}^h, \quad j \in [1, N_S] \end{aligned}, \quad (2)$$

$$\begin{aligned} &o_I^k = \texttt{BERT}\left(\texttt{[CLS]}\, n_I^k : d_I^k\, \texttt{[SEP]}\right) \\ &v_I^k = W_I \cdot o_I^{k,\texttt{[CLS]}} \in \mathbb{R}^h, \quad k \in [1, N_I] \end{aligned}. \quad (3)$$

N_S and N_I mean the number of slots and intents for the service, respectively. n_S^j is the j-th slot name, and d_S^j is the j-th slot description. $o_S^{j,\texttt{[CLS]}}$ is the output of `[CLS]` token from the j-th slot, and $W_S \in \mathbb{R}^{h \times h}$ is an FC layer. Similarly, n_I^k is the k-th intent name, and d_I^k is the k-th intent description. $o_I^{k,\texttt{[CLS]}}$ is the output of `[CLS]` token from the k-th intent, and $W_I \in \mathbb{R}^{h \times h}$ is an FC layer. The schema encoder takes v_R, V_S, and V_I to update the slot vectors V_S and intent vectors V_I

with attention mechanism as follows:

$$\begin{aligned} a_S &= \texttt{softmax}\left(V_S \cdot v_R\right) \in \mathbb{R}^{N_S} \\ v_{R,S} &= \left(V_S\right)^T \cdot a_S \in \mathbb{R}^h \end{aligned}, \tag{4}$$

$$v_S^j = W_{RS} \cdot \left(v_{R,S} \oplus v_S^j\right) \in \mathbb{R}^h, \tag{5}$$

$$\begin{aligned} a_I &= \texttt{softmax}\left(V_I \cdot v_R\right) \in \mathbb{R}^{N_I} \\ v_{R,I} &= \left(V_I\right)^T \cdot a_I \in \mathbb{R}^h \end{aligned}, \tag{6}$$

$$v_I^k = W_{RI} \cdot \left(v_{R,I} \oplus v_I^k\right) \in \mathbb{R}^h. \tag{7}$$

$W_{RS} \in \mathbb{R}^{h\times 2h}$ and $W_{RI} \in \mathbb{R}^{h\times 2h}$ are FC layers, and $\oplus$ means the concatenation of two vectors. The slot vectors and intent vectors updated with reference to the service vector are used for next steps: classification and generation.

3.2 Slot and Intent Classification

SET-DST takes the slot vectors and intent vectors to classify what slots and intents are activated by users. We use the pre-trained GPT-2[1] for the state generator that encodes the dialogue history and generates the dialogue state as a sequence of words. The state generator encodes the dialogue history H_t that is accumulated during the conversation and the previous dialogue state D_{t-1} to calculate the context vector C_t as

$$\begin{aligned} C_t &= \left\{c_t^1, \cdots, c_t^{N_C}\right\} \in \mathbb{R}^{N_C \times h} \\ &= \texttt{GPT-2}\left(D_{t-1} \oplus H_t\right), \end{aligned} \tag{8}$$

where $N_C = |C_t|$, and c_t^i means the GPT-2 output of the i-th word. Then, the last output of C_t is used to classify which slots and intents are activated in the current conversation as follows:

$$P\left(s_t^j = \texttt{Active}\right) = \mathcal{F}\left(c_t^{N_C}, v_S^j\right), \tag{9}$$

$$P\left(i_t^k = \texttt{Active}\right) = \mathcal{F}\left(c_t^{N_C}, v_I^k\right), \tag{10}$$

where $P(s_t^j = \texttt{Active})$ means the probability that the j-th slot is activated on turn t, and $P(i_t^k = \texttt{Active})$ means the probability that the k-th intent is activated on turn t. v_S^j and v_I^k indicate the slot vector of the j-th slot and the intent vector of the k-th intent, respectively, calculated by the schema encoder. $\mathcal{F}$ is a projection layer to calculate the probabilities using the context vector, slot vector, and intent vector. We define $\mathcal{F}(x, y)$ as a function transforming vectors x and y into a probability scalar as

$$\begin{aligned} h_1 &= \texttt{tanh}\left(W_1 \cdot x\right) \\ h_2 &= \texttt{tanh}\left(W_2 \cdot (h_1 \oplus y)\right), \\ \mathcal{F}(x, y) &= \sigma\left(W_3 \cdot h_2\right) \end{aligned} \tag{11}$$

where $W_1 \in \mathbb{R}^{h\times h}$, $W_2 \in \mathbb{R}^{h\times 2h}$, and $W_3 \in \mathbb{R}^{1\times h}$ are FC layers. Activate slots and intents are classified based on the probabilities $P(s_t^j = \texttt{Active})$ and $P(i_t^j = \texttt{Active})$. We define the slots activated on turn t as $S_t = \{s_t^j | P(s_t^j = \texttt{Active}) \geq \alpha,\ j \in [1, N_S]\}$ and the intents activated on turn t as $I_t = \{i_t^k | P(i_t^k = \texttt{Active}) \geq \alpha,\ k \in [1, N_I]\}$. s_t^j and i_t^k means the j-th slot name and the k-th intent name, respectively. α is a threshold to classify the slots and intents based on the probabilities. Activated slots S_t and intents I_t classified on this step are used to generate the dialogue state. On the generation step, S_t and I_t further contains slot vectors and intent vectors, which are calculated on the encoding step, in addition to the names in the text form. For example, $S_t = \{\texttt{restaurant\_name, price\_range}\}$ can be represented as

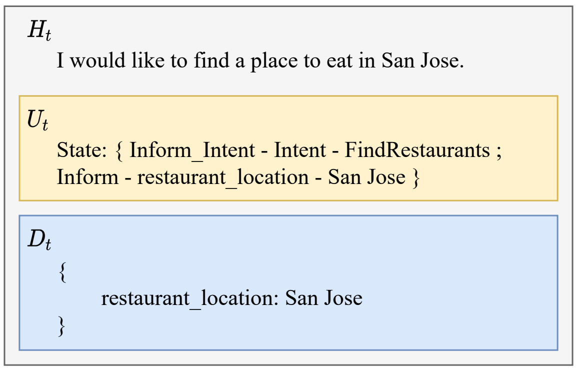
$$\begin{aligned} &E(\text{“}\texttt{Slots:\{restaurant\_name:}\text{”})\oplus \\ &v_S^1 \oplus E(\text{“}\texttt{;price\_range:}\text{”}) \oplus v_S^2 \oplus E(\text{“}\texttt{\}}\text{”}) \end{aligned} \tag{12}$$

before being fed into GPT-2, where E is the embedding layer to project the slot names into vector space of the same size as the slot vectors. To generate the dialogue state that consists of the slot-value pairs, the system should recognize not only the values but also the name of slots. The values can be extracted from user utterances, and which slots are activated can be predicted by the classification step; however, the exact slot names should be provided in the text form to construct slot-value structure matching given schema. By this process, the system can recognize the name of activated slots and intents before generating the dialogue state.

3.3 Dialogue State Generation

SET-DST has the state generator that generates dialogue state using the dialogue history, schema representation, and previous dialogue state accumulated during the conversation. In this paper, we define the dialogue state as a list of slot-value pairs that mean the details of an user goal. We also define the concept called user state that is a sequence

[1]The pre-trained models are available at https://github.com/huggingface/transformers.


H_t

I would like to find a place to eat in San Jose.

U_t

State: { Inform_Intent - Intent - FindRestaurants ; Inform - restaurant_location - San Jose }

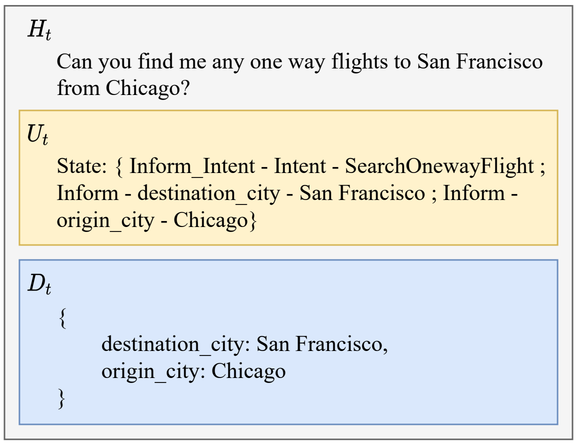
D_t

{

restaurant_location: San Jose

}

(a) Example in restaurants domain.


H_t

Can you find me any one way flights to San Francisco from Chicago?

U_t

State: { Inform_Intent - Intent - SearchOnewayFlight ; Inform - destination_city - San Francisco ; Inform - origin_city - Chicago}

D_t

{

destination_city: San Francisco,

origin_city: Chicago

}

(b) Example in flights domain.

Figure 3: Examples of user state and dialogue state corresponding to user utterance. U_t is a sequence of words, and D_t is a list of slot-value pairs.

of action-slot-value triples to generalize semantics from various user utterances. The state generator recurrently generates the user state as a sequence of words, instead of generating the dialogue state in the structured form directly. Then, the dialogue state is updated by extracting the slot-value pairs from the user state. The user state U_t on turn t is generated based on the previous dialogue state D_{t-1}, dialogue history H_t, active slots S_t, and active intents I_t as follows:

$$\tilde{u}_t^l = \texttt{GPT-2}\left(D_{t-1} \oplus H_t \oplus S_t \oplus I_t \oplus U_t^{1:l-1}\right), \tag{13}$$

$$\begin{aligned} U_t = \Big\{ u_t^l \Big| u_t^l = \texttt{argmax}\left(W_{vocab} \cdot \tilde{u}_t^l\right), \\ l \in [1, N_U] \Big\} \in \mathbb{R}^{N_U}, \end{aligned} \tag{14}$$

where $U_t^{1:l-1} = \{u_t^1, \cdots, u_t^{l-1}\}$ and $N_U = |U_t|$. u_t^l means the l-th word of the user state, and $W_{vocab} \in \mathbb{R}^{N_{vocab} \times h}$ is an FC layer to project the hidden state to vocabulary space with size of N_{vocab}. Figure 3 shows how to generate D_t from U_t. U_t is generated word-by-word over time steps until `[EOS]`, a special word to terminate the generation, is detected. Then, D_t is updated by extracting the slot-value pairs from U_t. In task-oriented dialogue system, the dialogue state is used to call API. In Figure 3a, `San Jose` is passed as the value of an argument named `restaurant_location` to call the API named `FindRestaurants`. However, we aim not to build the full task-oriented dialogue system but to generate the dialogue state in this study.

3.4 Optimization

SET-DST is optimized over two steps: (1) slot and intent classification, and (2) state generation. We freeze the pre-trained BERT during training to preserve the broad and general knowledge that is learned from large corpus. In classification task, the system is trained by using binary cross-entropy. Equation 9 is used to calculate the slot loss $\mathcal{L}_t^S$ with slot labels $Y_t^S = \{y_{t,1}^S, \cdots, y_{t,N_S}^S\}$ as

$$\begin{aligned} \mathcal{L}_t^S = &- \frac{1}{N_S} \sum_{j=1}^{N_S} \beta \cdot y_{t,j}^S \cdot \log P\left(s_t^j\right) \\ &+ \left(1 - y_{t,j}^S\right) \log\left(1 - P\left(s_t^j\right)\right), \end{aligned} \tag{15}$$

where $y_{t,j}^S \in \mathbb{R}^1$ is the binary value of j-th slot on turn t, and β is a hyperparameter to consider the ratio of active slots out of total slots. Based on Equation 10, the intent loss $\mathcal{L}_t^I$ is calculated with intent labels $Y_t^I = \{y_{t,1}^I, \cdots, y_{t,N_I}^I\}$ as

$$\begin{aligned} \mathcal{L}_t^I = &- \frac{1}{N_I} \sum_{k=1}^{N_I} \beta \cdot y_{t,k}^I \cdot \log P\left(i_t^k\right) \\ &+ \left(1 - y_{t,k}^I\right) \log\left(1 - P\left(i_t^k\right)\right), \end{aligned} \tag{16}$$

where $y_{t,k}^I \in \mathbb{R}^1$ is the binary value of k-th intent on turn t. In state generation step, the system is trained as a conditional language model that recurrently generates words over time steps. The state loss $\mathcal{L}_t^U$ is calculated base on Equation 14 with the state label $Y_t^U = \{y_{t,l}^U, \cdots, y_{t,N_U}^U\}$ as

$$\mathcal{L}_t^U = -\frac{1}{N_U} \sum_{l=1}^{N_U} \left(y_{t,l}^U\right)^T \log P\left(u_t^l\right), \tag{17}$$

where $y_{t,l}^U \in \mathbb{R}^{N_{vocab}}$ is the one-hot vector that indicates the l-th word of the gold-standard user state on turn t. The final joint loss is the sum of above losses:

$$\mathcal{L}_t = \mathcal{L}_t^S + \mathcal{L}_t^I + \mathcal{L}_t^U. \tag{18}$$

We use Adam optimizer (Kingma and Ba, 2014) to minimize $\mathcal{L}_t$.

4 Experiments

In this section, we describe our experiments including the datasets, evaluation metric, and results.

4.1 Experimental Setups

We used two datasets MultiWOZ 2.1[2] and Schema-Guided Dialogue (SGD)[3] to evaluate our system. MultiWOZ consists of conversations between a tourist and a guide, e.g., booking hotels and searching trains. SGD deals with conversations between a virtual assistant and an user ranging over various domains, e.g., events, restaurants, and media. The dataset also provides a schema that includes services, intents, and slots with short descriptions to help understanding the conversations. In this study, we followed the schema proposed in SGD. MultiWOZ has about 10,400 dialogues, and SGD has about 22,800 dialogues.

The datasets propose joint accuracy as the metric to evaluate DST systems. Joint accuracy measures whether a system successfully predicts all slot-value pairs mentioned on the conversations. In every turn, the system updates dialogue state, and the joint accuracy is calculated based on the accumulated dialogue state.

4.2 Experimental Details

The motivation of SET-DST is to make the system interpret the schema and refer it for efficiently tracking the dialogue state. In the experiments, our goal is to verify that SET-DST works well for our purpose by improving the performance of DST and the efficiency on few-shot settings with the schema encoding.

The experiments are divided into two steps: (1) pre-training on SGD and (2) fine-tuning on MultiWOZ. In the pre-training step, SET-DST is optimized to encode the schema for DST. In the fine-tuning step, the capability that encodes given

[2]https://github.com/budzianowski/multiwoz.

[3]https://github.com/google-research-datasets/dstc8-schema-guided-dialogue.

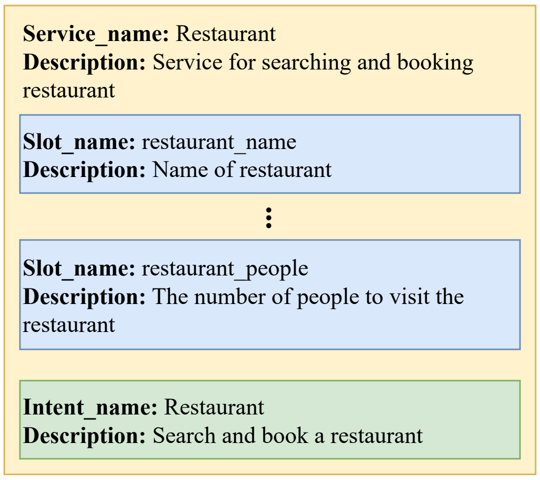
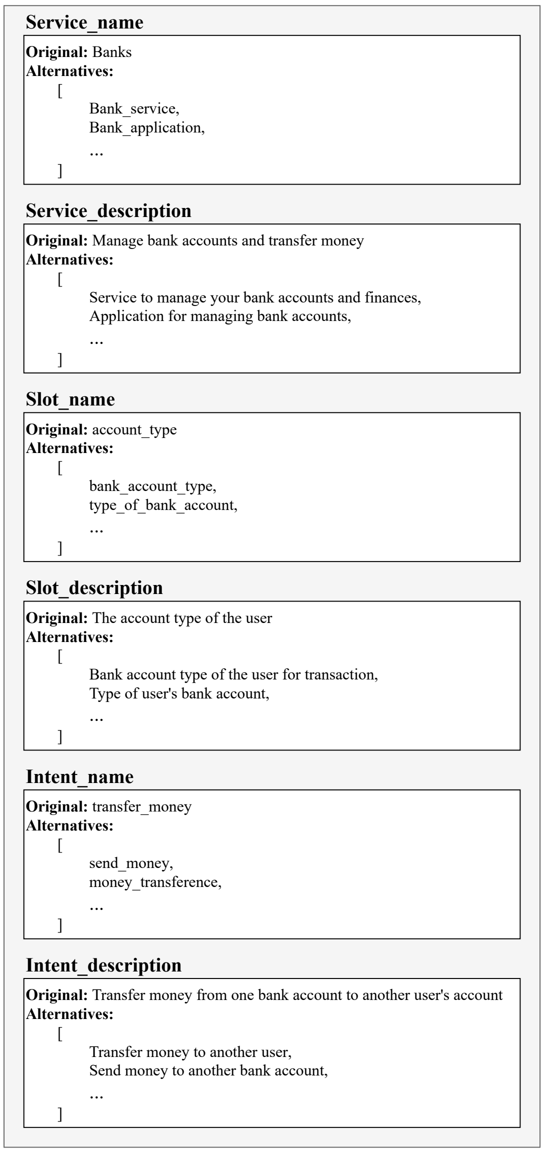

Service_name: Restaurant
Description: Service for searching and booking restaurant
Slot_name: restaurant_name
Description: Name of restaurant
⋮
Slot_name: restaurant_people
Description: The number of people to visit the restaurant
Intent_name: Restaurant
Description: Search and book a restaurant

Figure 4: Example of schema that is temporarily created for MultiWOZ dataset.

schema is transferred to encode new schema for improvement of the performance and efficiency. We conducted the experiments by adjusting the rate of few-shot data during fine-tuning to focus on the fine-tuning step. The training data for few-shot settings was randomly sampled from the training set of MultiWOZ, and the random seed was fixed for consistency of sampling. We also conducted experiments to verify whether SET-DST successfully works on the pre-training step, although the major part in our experiments is the fine-tuning on MultiWOZ including few-shot settings.

SET-DST needs not only slot information but also a schema. However, MultiWOZ has no schema and no concepts of service and intent; thus, we created a schema for MultiWOZ including services, slots, intents, and corresponding descriptions. Figure 4 shows an example of the schema for MultiWOZ. In our experiments on MultiWOZ, an intent means activated domain. In other words, the system classifies an intent as active when the domain of conversation is changed or a conversation starts. MultiWOZ further has no labels for activated intents, thus we automatically added the labels by tracking active domains and detecting whether new domains are active.

We further tried to fine-tune the system without intents because it is possible that the concepts of intent are unnatural in MultiWOZ. In this setting, Equation 3, 6, 7, 10, 16 are ignored, I_t is removed from Equation 13, and $\mathcal{L}_t^I$ is removed from Equation 18. In Figure 3a, U_t is replaced with `State: { Inform - restaurant_location - San Jose },`



Service_name

Original: Banks
Alternatives:
[
Bank_service,
Bank_application,
...
]

Service_description

Original: Manage bank accounts and transfer money
Alternatives:
[
Service to manage your bank accounts and finances,
Application for managing bank accounts,
...
]

Slot_name

Original: account_type
Alternatives:
[
bank_account_type,
type_of_bank_account,
...
]

Slot_description

Original: The account type of the user
Alternatives:
[
Bank account type of the user for transaction,
Type of user's bank account,
...
]

Intent_name

Original: transfer_money
Alternatives:
[
send_money,
money_transference,
...
]

Intent_description

Original: Transfer money from one bank account to another user's account
Alternatives:
[
Transfer money to another user,
Send money to another bank account,
...
]

Figure 5: Example of alternatives for schema augmentation.

and in Figure 3b, U_t is replaced with `State: { Inform - destination_city - San Francisco ; Inform - origin_city - Chicago }`.

A dataset schema can be variously defined depending on the developer, and our goal is to make the system represent any schema for DST. In the pre-training step, we manually augmented the schema of SGD dataset to avoid overfitting to the given schema. The schema provides names of services, slots, and intents with short descriptions. We defined some alternatives of the names and descriptions, and sampled inputs for the schema encoder from the augmented schema. Figure 5 shows some examples of the alternatives for bank service.

In dialogue state, multiple slots and intents can be activated at once. However, the order has no meaning in dialogue state. The state generator is trained to generate the dialogue state based on textual label, so it is possible that the order causes wrong optimization and overfitting. Thus, we shuffled the order of slots and intents when making the labels.

We used BERT-base-uncased[4] model for the schema encoder and GPT-2[5] model for the state generator. In our experiments, the pre-training step took about two days, and the fine-tuning step took about a day on a TitanRTX GPU. Table 3 lists hyperparameters that are used in our experiments.

	JA
TRADE (Wu et al., 2019)	45.60%*
DSTQA (Zhou and Small, 2019)	51.17%
LABES-S2S (Zhang et al., 2020b)	51.45%
DST-Picklist (Zhang et al., 2020a)	53.30%
MinTL-BART (Lin et al., 2020)	53.62%
TripPy (Heck et al., 2020)	55.29%
SimpleTOD (Hosseini-Asl et al., 2020)	55.76%
PPTOD (Su et al., 2021)	57.45%
ConvBERT-DG (Mehri et al., 2020)	58.70%
TripPy+SCoRe (Yu et al., 2020)	60.48%
TripPy+CoCoAug (Li et al., 2020)	60.53%
TripPy+SaCLog (Dai et al., 2021)	60.61%
SET-DST (Ours)	60.39%
SET-DST w/o intent	**62.07**%

Table 1: DST results on the test set of MultiWOZ in joint accuracy. *: the result is reported by Eric et al. (2020).

4.3 Experimental Results

Table 1 compares the evaluation results of SET-DST to the previous methods on the test set of MultiWOZ. In our experiments, SET-DST achieved new state-of-the-art joint accuracy when fine-tuned without intent.

Table 2 shows the evaluation results on few-shot settings and the improvement by pre-training. When we used less training data, the pre-training with schema encoding was more effective for DST. SET-DST performed reasonably well with only about 20% of the training data. In most cases, the models fine-tuned without intents achieved higher joint accuracy on MultiWOZ.

[4]`https://huggingface.co/bert-base-uncased`.

[5]`https://huggingface.co/gpt2`.

Few-shot rate		JA	
		w/ intent	w/o intent
w/ pre-training	100%	60.39%	62.07%
	30%	53.43%	56.43%
	25%	53.07%	55.61%
	20%	52.73%	54.37%
	15%	40.41%	51.29%
	10%	31.20%	29.91%
w/o pre-training	100%	58.37%	59.10%
	30%	31.08%	48.96%
	25%	30.39%	30.28%
	20%	22.80%	22.35%
	15%	19.38%	17.09%
	10%	10.21%	15.10%

Table 2: Evaluation results on few-shot settings with considering pre-training.

5 Discussion

Schema Encoding In this study, our goal is to transfer a pre-trained DST model to a low-resource domain without limiting the transference as language model level by using schema encoding. We pre-trained SET-DST on SGD which is a relatively large dataset and fine-tuned it on MultiWOZ to transfer the schema encoding. As a result, the pre-training significantly improved the accuracy on DST. Table 2 shows that the pre-training was more effective when the target dataset was small. To satisfy the joint accuracy evaluation, the system should not only predict values, but also exactly match the names of slots defined on the dataset. We believe that the tuning process was not completed when we used just 10% of the target dataset; on the other side, the system could match the slot names defined on MultiWOZ not SGD with 20% of the dataset. Pre-trained language models have been already used in many fields. However, our method could tackle general DST beyond language modeling on various domains. We believe that SET-DST can assist the development of DST systems in real world without large dataset on the target domain.

Intent on Fine-tuning In this paper, we define the intents as sub-goals to be achieved through a service. SGD has a schema for dialogues between a virtual assistant and an user. Thus, it is assumed that a system achieves the user's sub-goals by using APIs, and an intent corresponds to an API. Virtual assistant should tackle various services that could consist of one more intents, e.g., to check account balance and to transfer money in bank service. Unlike that, MultiWOZ has no schema and considers no APIs as intents. Thus, the schema that we temporarily created for experiments in the same form as the schema of SGD could cause confusion in generation of dialogue state. We believe that this is why the results without intent were slightly higher in the experiments. Another reason would be the incorrect labels for intents that we automatically created for experiments on MultiWOZ.

The joint accuracy that has been proposed as an evaluation metric for DST considers only slot-value pairs. However, task-oriented dialogue systems should call APIs of external systems to achieve goals, e.g., to search restaurants and to reserve hotels. The systems that predict only slot-value pairs would be insufficient to replace rule-based traditional systems in real-world. Even though use of intents made no improvement in joint accuracy, we believe that encoding the schema including intents is meaningful in terms of approaching more realistic DST.

Backbone Model We used BERT for schema encoding and GPT-2 for classification and generation. Using only GPT-2 was an option in our study. However, we fixed the pre-trained BERT during optimization to preserve the broad and general knowledge learned on pre-training and to use the knowledge for encoding given schema. BERT is an excellent backbone model to encode text for downstream tasks; thus, we used BERT instead of GPT-2 for the schema encoding.

6 Conclusion

Transfer learning that makes it possible to apply a pre-trained model to new domains has been attempted a lot. However, the attempts for DST have been just to use large-scale pre-trained models as language models. In this paper, we have proposed SET-DST, which is an effective method for DST with transfer learning by using schema encoding. We have demonstrated how to encode the schema for transferable DST and how to use the schema representation for dialogue state generation. Our experiments show that the schema encoding improves joint accuracy even in few-shot settings.

Even though our approach could perform DST well on target domain with few-shot settings, it required some new data to be fine-tuned. As part of our future work, we plan to design a DST model for zero-shot settings.

Acknowledgements

This work was supported by Institute for Information &communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00354, Artificial intelligence technology inferring issues and logically supporting facts from raw text), and by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2022-2020-0-01789) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

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.

Yinpei Dai, Hangyu Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, and Xiaodan Zhu. 2021. Preview, attend and review: Schema-aware curriculum learning for multi-domain dialog state tracking. *arXiv preprint arXiv:2106.00291*.

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

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. Efficient dialogue state tracking by selectively overwriting memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. Sumbt: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. 2020. Coco: Controllable counterfactuals for evaluating dialogue state trackers. *arXiv preprint arXiv:2010.12850*.

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.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Jiyoon Han, Jangwon Park, Chisung Song, Junseong Kim, Yongsook Song, Taehwan Oh, et al. 2021. Klue: Korean language understanding evaluation. *arXiv preprint arXiv:2105.09680*.

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*, 9:907–824.

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

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

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2786.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2021. Multi-task pre-training for plug-and-play task-oriented dialogue system. *arXiv preprint arXiv:2109.14739*.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432.

Jason D Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291.

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.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2020. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, S Yu Philip, Richard Socher, and Caiming Xiong. 2020a. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 154–167.

Yichi Zhang, Zhijian Ou, Min Hu, and Junlan Feng. 2020b. A probabilistic end-to-end task-oriented dialog model with latent belief states towards semi-supervised learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9207–9219.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.

Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020. Efficient context and schema fusion networks for multi-domain dialogue state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 766–781.

A Hyperparameters

We list the hyperparameters used in our experiments for reproducibility.

Hidden size	768
Embedding size	768
Vocabulary size	30522
Dropout	0.3
Early stopping count	5
Max epochs	40
Min epochs	20
Batch size	8
Learning rate	3e-5
Gradient clipping	10
α	0.5
β (on SGD)	3
β (on MultiWOZ)	5

Table 3: Hyperparameters used for the experiments in this paper.

B User Actions Set

Table 4 lists the user actions covered in this paper. When the user state U_t is generated, each element has three types: (1) action-slot-value triple, (2) action-slot pair, and (3) only action; e.g., (1) `Inform-restaurant_location-SanJose`, (2) `Request-restaurant_address`, and (3) `Negate`.

Action Name	Need Slot	Need Value
Inform	✓	✓
Inform_Intent*	✓	✓
Request	✓	×
Request_Alts	×	×
Affirm	×	×
Affirm_Intent*	✓	✓
Select	✓	✓
Negate	×	×
Negate_Intent	×	×
Thank_You	×	×
Goodbye	×	×

Table 4: List of user actions covered in this paper. * just need an intent as the value, but we added a dummy slot, `Intent`, to keep the shape of action-slot-value triple; e.g., we used `Inform_Intent-Intent-FindRestaurants` instead of `Inform_Intent-FindRestaurants` when we make the user state.

C Pre-traing Results

We evaluated the pre-training performance on KLUE[6] dataset (Park et al., 2021), which is a Korean dataset for DST, in addition to SGD. Table 5 shows the pre-training results on SGD and KLUE. SET-DST outperformed the baselines. These results demonstrate that SET-DST successfully performs DST with just pre-training. In the experiment on KLUE, we used KLUE-BERT[7] and KoGPT-2[8] that are large-scale language models pre-trained on Korean corpus.

		JA
SGD	Baseline (Rastogi et al., 2020)	25.40%
	SET-DST (Ours)	55.56%
KLUE	Baseline (Park et al., 2021)	50.22%
	SET-DST (Ours)	57.61%

Table 5: Pre-training results on SGD and KLUE compared to their baselines.

D Example of dialogues

We add an example of dialogue with the corresponding user state U_t and dialogue state D_t. Table 6 shows an dialogue on three domains: hotels, flights, and attractions.

[6] `https://github.com/KLUE-benchmark/KLUE`.

[7] `https://huggingface.co/klue/bert-base`.

[8] `https://huggingface.co/skt/kogpt2-base-v2`.

User:	I want to find me a 4 star hotel in Sydney.
U_t:	Inform_Intent - Intent - SearchHotel ; Inform - hotel_location - Sydney ; Inform - star_rating - 4
D_t:	{ **hotel_location: Sydney**, **star_rating: 4** }
System:	I found 10 option that you may like. A 4 star hotel called 28 Hotel Sydney.
User:	Can you smoke here? Tell me the address.
U_t:	Request - smoking_allowed ; Request - street_address
D_t:	{ hotel_location: Sydney, star_rating: 4 }
System:	No, smoking is not allowed here. 28 Regent Street, Chippendale New South Wales 2008, Australia is their address.
User:	Sounds okay. Find me a round trip flight to get there. I will start my trave on day after tomorrow and I want Premium Economy class tickets.
U_t:	Select - place_name - 28 Hotel Sydney ; Inform_Intent - Intent - SearchRoundtripFlights ; Inform - flights_departure_date - day after tomorrow ; Inform - seating_class - Premium Economy
D_t:	{ hotel_location: Sydney, star_rating: 4, **place_name: 28 Hotel Sydney**, **flights_departure_date: day after tomorrow**, **seating_class: Premium Economy** }
System:	On which day you will return?
User:	12th of March is the date on which I will return.
U_t:	Inform - return_date - 12th of March
D_t:	{ hotel_location: Sydney, star_rating: 4, place_name: 28 Hotel Sydney, flights_departure_date: day after tomorrow, seating_class: Premium Economy **return_date: 12th of March** }
System:	From which city you want to depart?
User:	From Las Vegas.
U_t:	Inform - origin_airport - Lag Vegas
D_t:	{ hotel_location: Sydney, star_rating: 4, place_name: 28 Hotel Sydney, flights_departure_date: day after tomorrow, seating_class: Premium Economy return_date: 12th of March, **origin_airport: Las Vegas** }
System:	I have 1 flight for you. You like United Airlines? The onward flight takes off at 4:30 am and return is take off at 4:55 pm. It has a layover and ticket cost $697.
User:	Good. Find me some attractions there that have free entry and is child-friendly.
U_t:	Inform_Intent - Intent - FindAttractions ; Inform - free_entry - True ; Inform - good_for_kids - True ; Inform - category - dontcare
D_t:	{ hotel_location: Sydney, star_rating: 4, place_name: 28 Hotel Sydney, flights_departure_date: day after tomorrow, seating_class: Premium Economy return_date: 12th of March, origin_airport: Las Vegas, **free_entry: True** **good_for_kids: True**, **category: dontcare** }
System:	You can check out a Sports Venue called ANZ Stadium.
User:	Good. Tell me their phone number.
U_t:	Select - attraction_name - ANZ Stadium ; Request - phone_number
D_t:	{ hotel_location: Sydney, star_rating: 4, place_name: 28 Hotel Sydney, flights_departure_date: day after tomorrow, seating_class: Premium Economy return_date: 12th of March, origin_airport: Las Vegas, free_entry: True good_for_kids: True, category: dontcare, **attraction_name: ANZ Stadium** }
System:	2 9298 3777 is the phone number.
User:	Great. That's all that I wanted for now. Bye.
U_t:	GoodBye

Table 6: Example of dialogue including the user state and dialogue state that we defined in this paper.