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 trained to imitate the given dataset. In this paper, we propose Schema Encoding for Transferable Dialogue State Tracking (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. As a result, SET-DST improved the 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 to classify the users’ intents and 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 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 can 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 Schema Encoding for Transferable Dialogue State Tracking (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. 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 evaluated SET-DST on MultiWOZ 2.1 (Eric et al., 2020), which is a standard benchmark dataset for DST, as a downstream task. SET-DST achieved state-of-the-art accuracy on the DST task. We further confirmed that SET-DST worked well when
the size of the downstream dataset was small. 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 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 What services the system provides to users. It is similar to domain, but smaller and
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.

**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.

**Value** The values that have actual meaning for 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 user goals by conversation. 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.

### 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

\[
o_R = \text{BERT} ([\text{CLS}] n_R : d_R [\text{SEP}]), \quad v_R = W_R \cdot o_R^{\text{CLS}} \in \mathbb{R}^h,
\]

where \(n_R\) is the service name, \(d_R\) is the service description, and \(h\) is the hidden size. \(o_R^{\text{CLS}}\) is the output of \([\text{CLS}]\) token, and \(W_R \in \mathbb{R}^{h \times h}\) is a fully connected (FC) layer. \([\text{CLS}]\) and \([\text{SEP}]\) are special tokens that mean the start and end of the sentence, respectively. The service in Figure 2 can be represented as

<table>
<thead>
<tr>
<th>Service_name: Restaurants_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: A leading provider for restaurant search and reservations</td>
</tr>
<tr>
<td>Slot_name: restaurant_name</td>
</tr>
<tr>
<td>Description: Name of the restaurant</td>
</tr>
<tr>
<td>Slot_name: price_range</td>
</tr>
<tr>
<td>Description: Price range for the restaurant</td>
</tr>
<tr>
<td>Intent_name: ReserveRestaurant</td>
</tr>
<tr>
<td>Description: Reserve a table at a restaurant</td>
</tr>
<tr>
<td>Intent_name: FindRestaurants</td>
</tr>
<tr>
<td>Description: Find a restaurant of a particular cuisine in a city</td>
</tr>
</tbody>
</table>

![Figure 2: Example of schema for restaurant search and reservation service including slots and intents.](image-url)
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

\[
C_t = \{c_1, \ldots, c_{N_C}\} \in \mathbb{R}^{N_C \times h}
\]

\[
= \text{GPT-2} \left( D_{t-1} \oplus H_t \right),
\]

where \(N_C = |C_t|\). Then, the last output of \(C_t\) is used to classify which slots and intents are activated in the current conversation as follows:

\[
P \left( s^j_t = \text{Active} \right) = \mathcal{F} \left( c^{N_C}_{t}, v^j_S \right),
\]

\[
P \left( i^k_t = \text{Active} \right) = \mathcal{F} \left( c^{N_C}_{t}, v^k_I \right),
\]

where \(P(s^j_t = \text{Active})\) means the probability that the \(j\)-th slot is activated on turn \(t\), and \(P(i^k_t = \text{Active})\) means the probability that the \(k\)-th intent is activated on turn \(t\). \(\mathcal{F}\) is a projection layer to calculate the probabilities using the context vector, slot vector, and intent vector. We define \(P = \mathcal{F}(x, y)\) as a function transforming vectors \(x\) and \(y\) into a probability scalar as

\[
h_1 = \tanh \left( W_1 \cdot x \right)
\]

\[
h_2 = \tanh \left( W_2 \cdot (h_1 \oplus y) \right),
\]

\[
P = \sigma \left( W_3 \cdot h_2 \right)
\]

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 \(P(s^j_t = \text{Active})\) and \(P(i^k_t = \text{Active})\). We define the slots activated on turn \(t\) as \(S_t = \{s^j_t | P(s^j_t = \text{Active}) \geq \alpha, j \in [1, N_S]\}\) and the intents activated on turn \(t\) as \(I_t = \{i^k_t | P(i^k_t = \text{Active}) \geq \alpha, k \in [1, N_I]\}\), \(\alpha\) is a threshold to classify the slots and intents based on the probabilities.

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 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, and updates the dialogue state 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:

\[
\bar{u}^l_t = \text{GPT-2} \left( D_{t-1} \oplus H_t \oplus S_t \oplus I_t \oplus U^{l-1}_t \right),
\]

\[
U_t = \left\{ u^l_t | u^l_t = \text{argmax} \left( W_{vocab} \cdot \bar{u}^l_t \right), \ l \in [1, N_U] \right\} \in \mathbb{R}^{N_U},
\]

\footnote{The pre-trained models are available at \url{https://github.com/huggingface/transformers}.}
where $U^{1:t-1}_t = \{u^1_t, \cdots, u^{t-1}_t\}$, $N_U = |U_t|$, and $W_{\text{voc}} \in \mathbb{R}^{N_{\text{voc}} \times h}$ is an FC layer to project the hidden state to vocabulary space with size of $N_{\text{voc}}$. Figure 3 shows how to generate $D_t$ from $U_t$. $U_t$ is generated word-by-word over time steps until $[\text{EOS}]$, a special word to terminate the generation, is detected. Then, $D_t$ is updated by extracting the slot-value pairs from $U_t$.

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 $L^S_t$ with slot labels $Y^S_t = \{y^S_{t,1}, \cdots, y^S_{t,N_S}\}$ as

$$L^S_t = -\frac{1}{N_S} \sum_{j=1}^{N_S} \beta \cdot y^S_{t,j} \cdot \log P(s^j_t) + (1 - y^S_{t,j}) \log (1 - P(s^j_t)),$$  

(14)

where $y^S_{t,j} \in \mathbb{R}^1$ is the binary value of $j$-th slot on turn $t$, and $\beta$ is a hyperparameter to consider the ratio of active slots out of total slots. Based on Equation 10, the intent loss $L^I_t$ is calculated with intent labels $Y^I_t = \{y^I_{t,1}, \cdots, y^I_{t,N_I}\}$ as

$$L^I_t = -\frac{1}{N_I} \sum_{k=1}^{N_I} \beta \cdot y^I_{t,k} \cdot \log P(i^k_t) + (1 - y^I_{t,k}) \log (1 - P(i^k_t)),$$  

(15)

where $y^I_{t,k} \in \mathbb{R}^1$ 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 $L^U_t$ is calculated base on Equation 13 with the state label $Y^U_t = \{y^U_{t,1}, \cdots, y^U_{t,N_U}\}$ as

$$L^U_t = -\frac{1}{N_U} \sum_{l=1}^{N_U} (y^U_{t,l})^T \log P(u^l_t),$$  

(16)

where $y^U_{t,l} \in \mathbb{R}^{N_{\text{voc}}}$ 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:

$$L_t = L^S_t + L^I_t + L^U_t.$$  

(17)

We use Adam optimizer (Kingma and Ba, 2014) to minimize $L_t$.

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

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\footnote{https://github.com/budzianowski/multiwoz.} and Schema-Guided Dialogue (SGD)\footnote{https://github.com/google-research-datasets/dstc8-schema-guided-dialogue.} 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. 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 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 the pre-training to transfer the schema encoding improves the performance and 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. We evaluated the pre-training performance on KLUE dataset (Park et al., 2021), which is a Korean dataset for DST, in addition to SGD.

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 detecting the changes of domain.

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, 15 are ignored, \( I_t \) is removed from Equation 12, and \( L^I_t \) is removed from Equation 17. In Figure 3a, \( U_t \) is replaced with State: { Inform - restaurant_location - San Jose }, and in Figure 3b, \( U_t \) is replaced with State: 

4https://github.com/KLUE-benchmark/KLUE.

<table>
<thead>
<tr>
<th>Service_name</th>
<th>Original:</th>
<th>Banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives:</td>
<td>Bank_service, Bank_application</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service_description</th>
<th>Original:</th>
<th>Manage bank accounts and transfer money</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives:</td>
<td>Service to manage your bank accounts and finances, Application for managing bank accounts</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slot_name</th>
<th>Original:</th>
<th>account_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives:</td>
<td>bank_account_type, type_of_bank_account</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slot_description</th>
<th>Original:</th>
<th>The account type of the user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives:</td>
<td>Bank account type of the user for transaction, Type of user's bank account</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intent_name</th>
<th>Original:</th>
<th>transfer_money</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives:</td>
<td>send_money, money_transference</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intent_description</th>
<th>Original:</th>
<th>Transfer money from one bank account to another user's account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternatives:</td>
<td>Transfer money to another user, Send money to another bank account</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Example of alternatives for schema augmentation.

| 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 |
| \( \alpha \) | 0.5 |
| \( \beta \) (on SGD) | 3 |
| \( \beta \) (on MultiWOZ) | 5 |

Table 1: Hyperparameters used for the experiments in this paper.
Table 2 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 3 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.

Table 4 shows the pre-training results on SGD and KLUE compared to their baselines.

4.3 Experimental Results

Table 2 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 3 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.

Table 4 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\textsuperscript{7} and KoGPT-2\textsuperscript{8} that are large-scale language models pre-trained on Korean corpus.

\textsuperscript{5}https://huggingface.co/bert-base-uncased.
\textsuperscript{6}https://huggingface.co/gpt2.
\textsuperscript{7}https://huggingface.co/klue/bert-base.
\textsuperscript{8}https://huggingface.co/skt/kogpt2-base-v2.
5 Discussion

Schema Encoding Development of training dataset for task-oriented dialogue systems including DST requires a lot of cost, especially in low-resource language. Thus, recently, pre-trained models have been widely used to improve the efficiency of training by transferring the knowledge that is learned on large corpus. 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. 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. 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.

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. 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.

References


