Efficient Context and Schema Fusion Networks for Multi-Domain Dialogue State Tracking

Su Zhu, Jieyu Li, Lu Chen* and **Kai Yu***
State Key Laboratory of Media Convergence Production Technology and Systems
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{paul2204,oracion,chenlusz,kai.yu}@sjtu.edu.cn

Abstract

Dialogue state tracking (DST) aims at estimating the current dialogue state given all the preceding conversation. For multi-domain DST, the data sparsity problem is a major obstacle due to increased numbers of state candidates and dialogue lengths. To encode the dialogue context efficiently, we utilize the previous dialogue state (predicted) and the current dialogue utterance as the input for DST. To consider relations among different domain-slots, the schema graph involving prior knowledge is exploited. In this paper, a novel context and schema fusion network is proposed to encode the dialogue context and schema graph by using internal and external attention mechanisms. Experiment results show that our approach can outperform strong baselines, and the previous state-of-the-art method (SOM-DST) can also be improved by our proposed schema graph.

1 Introduction

Dialogue state tracking (DST) is a key component in task-oriented dialogue systems which cover certain narrow domains (e.g., *booking hotel* and *travel planning*). As a kind of context-aware language understanding task, DST aims to extract user goals or intents hidden in human-machine conversation and represent them as a compact dialogue state, i.e., a set of slots and their corresponding values. For example, as illustrated in Fig. 1, (*slot*, *value*) pairs like (*name*, *huntingdon marriott hotel*) are extracted from the dialogue. It is essential to build an accurate DST for dialogue management (Young et al., 2013), where dialogue state determines the next machine action and response.

Recently, motivated by the tremendous growth of commercial dialogue systems like Apple Siri, Microsoft Cortana, Amazon Alexa, or Google Assistant, multi-domain DST becomes crucial to help users across different domains (Budzianowski et al., 2018; Eric et al., 2019). As shown in Fig. 1, the dialogue covers three domains (i.e., `Hotel`, `Attraction` and `Taxi`). The goal of multi-domain DST is to predict the value (including `NONE`) for each *domain-slot* pair based on all the preceding dialogue utterances. However, due to increasing numbers of dialogue turns and domain-slot pairs, the data sparsity problem becomes the main issue in this field.

Figure 1: An example of multi-domain dialogues. Utterances at the left side are from the system agent, and utterances at the right side are from a user. The dialogue state of each domain is represented as a set of (*slot*, *value*) pairs.

To tackle the above problem, we emphasize that DST models should support open-vocabulary based value decoding, encode context efficiently and incorporate domain-slot relations:

1. Open-vocabulary DST is essential for real-world applications (Wu et al., 2019; Gao et al., 2019; Ren et al., 2019), since value sets for some slots can be very huge and variable (e.g., *song names*).

2. To encode the dialogue context efficiently, we

*The corresponding authors are Lu Chen and Kai Yu.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 766–781
November 16 - 20, 2020.

attempt to get context representation from the previous (predicted) dialogue state and the current turn dialogue utterance, while not concatenating all the preceding dialogue utterances.

3. To consider relations among domains and slots, we introduce the schema graph which contains *domain*, *slot*, *domain-slot* nodes and their relationships. It is a kind of prior knowledge and may help alleviate the data imbalance problem.

To this end, we propose a multi-domain dialogue state tracker with context and schema fusion networks (CSFN-DST). The fusion network is exploited to jointly encode the previous dialogue state, the current turn dialogue and the schema graph by internal and external attention mechanisms. After multiple layers of attention networks, the final representation of each *domain-slot* node is utilized to predict the corresponding value, involving context and schema information. For the value prediction, a slot gate classifier is applied to decide whether a domain-slot is mentioned in the conversation, and then an RNN-based value decoder is exploited to generate the corresponding value.

Our proposed CSFN-DST is evaluated on MultiWOZ 2.0 and MultiWOZ 2.1 benchmarks. Ablation study on each component further reveals that both context and schema are essential. Contributions in this work are summarized as:

- To alleviate the data sparsity problem and enhance the context encoding, we propose exploiting domain-slot relations within the schema graph for open-vocabulary DST.
- To fully encode the schema graph and dialogue context, fusion networks are introduced with graph-based, internal and external attention mechanisms.
- Experimental results show that our approach surpasses strong baselines, and the previous state-of-the-art method (SOM-DST) can also be improved by our proposed schema graph.

2 Related Work

Traditional DST models rely on semantics extracted by natural language understanding to predict the current dialogue states (Young et al., 2013; Williams et al., 2013; Henderson et al., 2014d; Sun et al., 2014b,a; Yu et al., 2015), or jointly learn language understanding in an end-to-end way (Henderson et al., 2014b,c). These methods heavily rely on hand-crafted features and complex domain-specific lexicons for delexicalization, which are difficult to extend to new domains. Recently, most works about DST focus on encoding dialogue context with deep neural networks (such as CNN, RNN, LSTM-RNN, etc.) and predicting a value for each possible slot (Mrkšić et al., 2017; Xu and Hu, 2018; Zhong et al., 2018; Ren et al., 2018).

Multi-domain DST Most traditional state tracking approaches focus on a single domain, which extract value for each slot in the domain (Williams et al., 2013; Henderson et al., 2014a). They can be directly adapted to multi/mixed-domain conversations by replacing slots in a single domain with *domain-slot* pairs (i.e. domain-specific slots) (Ramadan et al., 2018; Gao et al., 2019; Wu et al., 2019; Zhang et al., 2019; Kim et al., 2019). Despite its simplicity, this approach for multi-domain DST extracts value for each domain-slot independently, which may fail to capture features from slot co-occurrences. For example, hotels with higher *stars* are usually more expensive (*price_range*).

Predefined ontology-based DST Most of the previous works assume that a predefined ontology is provided in advance, i.e., all slots and their values of each domain are known and fixed (Williams, 2012; Henderson et al., 2014a). Predefined ontology-based DST can be simplified into a value classification task for each slot (Henderson et al., 2014c; Mrkšić et al., 2017; Zhong et al., 2018; Ren et al., 2018; Ramadan et al., 2018; Lee et al., 2019). It has the advantage of access to the known candidate set of each slot, but these approaches may not be applicable in the real scenario. Since a full ontology is hard to obtain in advance (Xu and Hu, 2018), and the number of possible slot values could be substantial and variable (e.g., *song names*), even if a full ontology exists (Wu et al., 2019).

Open-vocabulary DST Without a predefined ontology, some works choose to directly generate or extract values for each slot from the dialogue context, by using the encoder-decoder architecture (Wu et al., 2019) or the pointer network (Gao et al., 2019; Ren et al., 2019; Le et al., 2020). They can improve the scalability and robustness to unseen slot values, while most of them are not efficient in context encoding since they encode all the previous utterances at each dialogue turn. Notably, a multi-

domain dialogue could involve quite a long history, e.g., MultiWOZ dataset (Budzianowski et al., 2018) contains about 13 turns per dialogue on average.

Graph Neural Network Graph Neural Network (GNN) approaches (Scarselli et al., 2009; Veličković et al., 2018) aggregate information from graph structure and encode node features, which can learn to reason and introduce structure information. Many GNN variants are proposed and also applied in various NLP tasks, such as text classification (Yao et al., 2019), machine translation (Marcheggiani et al., 2018), dialogue policy optimization (Chen et al., 2018, 2019) etc. We introduce graph-based multi-head attention and fusion networks for encoding the schema graph.

3 Problem Formulation

In a multi-domain dialogue state tracking problem, we assume that there are M domains (e.g. *taxi*, *hotel*) involved, $\mathcal{D} = \{d_1, d_2, \cdots, d_M\}$. Slots included in each domain $d \in \mathcal{D}$ are denoted as a set $\mathcal{S}^d = \{s_1^d, s_2^d, \cdots, s_{|\mathcal{S}^d|}^d\}$.[1] Thus, there are J possible *domain-slot* pairs totally, $\mathcal{O} = \{O_1, O_2, \cdots, O_J\}$, where $J = \sum_{m=1}^{M} |\mathcal{S}^{d_m}|$. Since different domains may contain a same slot, we denote all distinct N slots as $\mathcal{S} = \{s_1, s_2, \cdots, s_N\}$, where $N \leq J$.

A dialogue can be formally represented as $\{(A_1, U_1, B_1), (A_2, U_2, B_2), \cdots, (A_T, U_T, B_T)\}$, where A_t is what the agent says at the t-th turn, U_t is the user utterance at t turn, and B_t denotes the corresponding dialogue state. A_t and U_t are word sequences, while B_t is a set of *domain-slot-value* triplets, e.g., (*hotel*, *price_range*, *expensive*). Value v_{tj} is a word sequence for j-th *domain-slot* pair at the t-th turn. The goal of DST is to correctly predict the value for each *domain-slot* pair, given the dialogue history.

Most of the previous works choose to concatenate all words in the dialogue history, $[A_1, U_1, A_2, U_2, \cdots, A_t, U_t]$, as the input. However, this may lead to increased computation time. In this work, we propose to utilize only the current dialogue turn A_t, U_t and the previous dialogue state B_{t-1} to predict the new state B_t. During the training, we use the ground truth of B_{t-1}, while the previous predicted dialogue state would be used in the inference stage.

Schema Graph To consider relations between different *domain-slot* pairs and exploit them as an additional input to guide the context encoding, we formulate them as a schema graph $G = (V, E)$ with node set V and edge set E. Fig. 2 shows an example of schema graph. In the graph, there are three kinds of nodes to denote all domains $\mathcal{D}$, slots $\mathcal{S}$, and domain-slot pairs $\mathcal{O}$, i.e., $V = \mathcal{D} \cup \mathcal{S} \cup \mathcal{O}$. Four types of undirected edges between different nodes are exploited to encode prior knowledge:

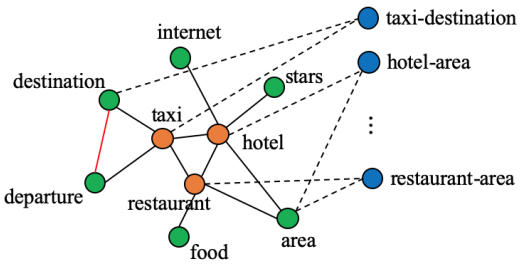

Figure 2: An example of schema graph. Domain nodes are in orange, slot nodes are in green and domain-slot nodes are in blue.

1. (d, d'): Any two domain nodes, $d \in \mathcal{D}$ and $d' \in \mathcal{D}$, are linked to each other.
2. (s, d): We add an edge between slot $s \in \mathcal{S}$ and domain $d \in \mathcal{D}$ nodes if $s \in \mathcal{S}^d$, .
3. (d, o) and (s, o): If a domain-slot pair $o \in \mathcal{O}$ is composed of the domain $d \in \mathcal{D}$ and slot $s \in \mathcal{S}$, there are two edges from d and s to this domain-slot node respectively.
4. (s, s'): If the candidate values of two different slots ($s \in \mathcal{S}$ and $s' \in \mathcal{S}$) would overlap, there is also an edge between them, e.g., *destination* and *departure*, *leave_at* and *arrive_by*.

4 Context and Schema Fusion Networks for Multi-domain DST

In this section, we will introduce our approach for multi-domain DST, which jointly encodes the current dialogue turn (A_t and U_t), the previous dialogue state B_{t-1} and the schema graph G by fusion networks. After that, we can obtain context-aware and schema-aware node embeddings for all J domain-slot pairs. Finally, a slot-gate classifier and RNN-based value decoder are exploited to extract the value for each domain-slot pair.

The architecture of CSFN-DST is illustrated in Fig. 3, which consists of input embeddings, context schema fusion network and state prediction modules.

[1] For open-vocabulary DST, possible values for each slot $s \in \mathcal{S}^d$ are not known in advance.

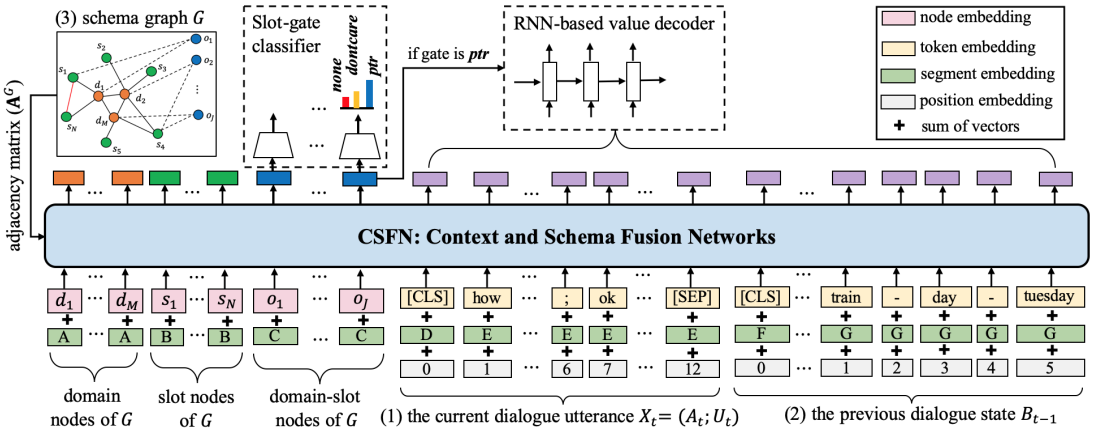

Figure 3: The overview of the proposed CSFN-DST. It takes the current dialogue utterance, the previous dialogue state and the schema graph as the input and predicts the current dialogue state. It consists of an embedding layer, context and schema fusion networks, a slot-gate classifier and an RNN-based value decoder.

4.1 Input Embeddings

Besides token and position embeddings for encoding literal information, segment embeddings are also exploited to discriminate different types of input tokens.

(1) Dialogue Utterance We denote the representation of the dialogue utterances at t-th turn as a joint sequence, $X_t = \texttt{[CLS]} \oplus A_t \oplus ; \oplus U_t \oplus \texttt{[SEP]}$, where `[CLS]` and `[SEP]` are auxiliary tokens for separation, $\oplus$ is the operation of sequence concatenation. As `[CLS]` is designed to capture the sequence embedding, it has a different segment type with the other tokens. The input embeddings of X_t are the sum of the token embeddings, the segmentation embeddings and the position embeddings (Vaswani et al., 2017), as shown in Fig. 3.

(2) Previous Dialogue State As mentioned before, a dialogue state is a set of *domain-slot-value* triplets with a mentioned value (not `NONE`). Therefore, we denote the previous dialogue state as $B_{t-1} = \texttt{[CLS]} \oplus R^1_{t-1} \oplus \cdots \oplus R^K_{t-1}$, where K is the number of triplets in B_{t-1}. Each triplet $d\text{-}s\text{-}v$ is denoted as a sub-sequence, i.e., $R = d \oplus - \oplus s \oplus - \oplus v$. The domain and slot names are tokenized, e.g., *price_range* is replaced with "price range". The value is also represented as a token sequence. For the special value `DONTCARE` which means users do not care the value, it would be replaced with "dont care". The input embeddings of B_{t-1} are the sum of the token, segmentation and position embeddings. Positions are re-enumerated for different triplets.

(3) Schema Graph As mentioned before, the schema graph G is comprised of M domain nodes, N slot nodes and J domain-slot nodes. These nodes are arranged as $G = d_1 \oplus \cdots \oplus d_M \oplus s_1 \oplus \cdots \oplus s_N \oplus o_1 \oplus \cdots \oplus o_J$. Each node embedding is initialized by averaging embeddings of tokens in the corresponding domain/slot/domain-slot. Positions embeddings are omitted in the graph. The edges of the graph are represented as an adjacency matrix $\mathbf{A}^G$ whose items are either one or zero, which would be used in the fusion network. To emphasize edges between different types of nodes can be different in the computation, we exploit node types to get segment embeddings.

4.2 Context and Schema Fusion Network

At this point, we have input representations $\mathrm{H}_0^G \in \mathbb{R}^{|G| \times d_m}$, $\mathrm{H}_0^{X_t} \in \mathbb{R}^{|X_t| \times d_m}$, $\mathrm{H}_0^{B_{t-1}} \in \mathbb{R}^{|B_{t-1}| \times d_m}$, where $|.|$ gets the token or node number. The context and schema fusion network (CSFN) is utilized to compute hidden states for tokens or nodes in X_t, B_{t-1} and G layer by layer. We then apply a stack of L context- and schema-aware self-attention layers to get final hidden states, $\mathrm{H}_L^G, \mathrm{H}_L^{X_t}, \mathrm{H}_L^{B_{t-1}}$. The i-th layer ($0 \le i < L$) can be formulated as:

$$\mathrm{H}_{i+1}^G, \mathrm{H}_{i+1}^{X_t}, \mathrm{H}_{i+1}^{B_{t-1}} = \text{CSFNLayer}_i(\mathrm{H}_i^G, \mathrm{H}_i^{X_t}, \mathrm{H}_i^{B_{t-1}})$$

4.2.1 Multi-head Attention

Before describing the fusion network, we first introduce the multi-head attention (Vaswani et al., 2017) which is a basic module. The multi-head attention can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is

computed by a compatibility function of the query with the corresponding key.

Consider a source sequence of vectors $Y = \{\boldsymbol{y}_i\}_{i=1}^{|Y|}$ where $\boldsymbol{y}_i \in \mathbb{R}^{1\times d_{\text{model}}}$ and $Y \in \mathbb{R}^{|Y|\times d_{\text{model}}}$, and a target sequence of vectors $Z = \{\boldsymbol{z}_i\}_{i=1}^{|Z|}$ where $\boldsymbol{z}_i \in \mathbb{R}^{1\times d_{\text{model}}}$ and $Z \in \mathbb{R}^{|Z|\times d_{\text{model}}}$. For each vector $\boldsymbol{y}_i$, we can compute an attention vector $\boldsymbol{c}_i$ over Z by using H heads as follows:

$$e_{ij}^{(h)} = \frac{(\boldsymbol{y}_i W_Q^{(h)})(\boldsymbol{z}_j W_K^{(h)})^\top}{\sqrt{d_{\text{model}}/H}};\ a_{ij}^{(h)} = \frac{\exp(e_{ij}^{(h)})}{\sum_{l=1}^{|Z|}\exp(e_{il}^{(h)})}$$

$$\boldsymbol{c}_i^{(h)} = \sum_{j=1}^{|Z|} a_{ij}^{(h)}(\boldsymbol{z}_j W_V^{(h)}); \boldsymbol{c}_i = \text{Concat}(\boldsymbol{c}_i^{(1)}, \cdots, \boldsymbol{c}_i^{(H)})W_O$$

where $1 \le h \le H$, $W_O \in \mathbb{R}^{d_{\text{model}}\times d_{\text{model}}}$, and $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_{\text{model}}\times(d_{\text{model}}/H)}$. We can compute $\boldsymbol{c}_i$ for every $\boldsymbol{y}_i$ and get a transformed matrix $C \in \mathbb{R}^{|Y|\times d_{\text{model}}}$. The entire process is denoted as a mapping MultiHead_Θ:

$$C = \text{MultiHead}_\Theta(Y, Z) \tag{1}$$

Graph-based Multi-head Attention To apply the multi-head attention on a graph, the graph adjacency matrix $\mathbf{A} \in \mathbb{R}^{|Y|\times|Z|}$ is involved to mask nodes/tokens unrelated, where $\mathbf{A}_{ij} \in \{0, 1\}$. Thus, $e_{ij}^{(h)}$ is changed as:

$$e_{ij}^{(h)} = \begin{cases} \frac{(\boldsymbol{y}_i W_Q^{(h)})(\boldsymbol{z}_j W_K^{(h)})^\top}{\sqrt{d_{\text{model}}/H}}, & \text{if } \mathbf{A}_{ij} = 1 \\ -\infty, & \text{otherwise} \end{cases}$$

and Eqn. (1) is modified as:

$$C = \text{GraphMultiHead}_\Theta(Y, Z, \mathbf{A}) \tag{2}$$

Eqn. (1), can be treated as a special case of Eqn. (2) that the graph is fully connected, i.e., $\mathbf{A} = \mathbf{1}$.

4.2.2 Context- and Schema-Aware Encoding

Each layer of CSFN consists of internal and external attentions to incorporate different types of inputs. The hidden states of the schema graph G at the i-the layer are updated as follows:

$$\begin{aligned} \mathrm{I}_{\text{GG}} &= \text{GraphMultiHead}_{\Theta_{\text{GG}}}(\mathrm{H}_i^G, \mathrm{H}_i^G, \mathbf{A}^G) \\ \mathrm{E}_{\text{GX}} &= \text{MultiHead}_{\Theta_{\text{GX}}}(\mathrm{H}_i^G, \mathrm{H}_i^{X_t}) \\ \mathrm{E}_{\text{GB}} &= \text{MultiHead}_{\Theta_{\text{GB}}}(\mathrm{H}_i^G, \mathrm{H}_i^{B_{t-1}}) \\ \mathrm{C}_G &= \text{LayerNorm}(\mathrm{H}_i^G + \mathrm{I}_{\text{GG}} + \mathrm{E}_{\text{GX}} + \mathrm{E}_{\text{GB}}) \\ \mathrm{H}_{i+1}^G &= \text{LayerNorm}(\mathrm{C}_G + \text{FFN}(\mathrm{C}_G)) \end{aligned}$$

where $\mathbf{A}^G$ is the adjacency matrix of the schema graph and LayerNorm(.) is layer normalization function (Ba et al., 2016). $\text{FFN}(x)$ is a feed-forward network (FFN) function with two fully-connected layer and an ReLU activation in between, i.e., $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$.

Similarly, more details about updating $\mathrm{H}_i^{X_t}, \mathrm{H}_i^{B_{t-1}}$ are described in Appendix A.

The context and schema-aware encoding can also be simply implemented as the original transformer (Vaswani et al., 2017) with graph-based multi-head attentions.

4.3 State Prediction

The goal of state prediction is to produce the next dialogue state B_t, which is formulated as two stages: 1) We first apply a slot-gate classifier for each domain-slot node. The classifier makes a decision among {NONE, DONTCARE, PTR}, where NONE denotes that a domain-slot pair is not mentioned at this turn, DONTCARE implies that the user can accept any values for this slot, and PTR represents that the slot should be processed with a value. 2) For domain-slot pairs tagged with PTR, we further introduced an RNN-based value decoder to generate token sequences of their values.

4.3.1 Slot-gate Classification

We utilize the final hidden vector of j-th domain-slot node in G for the slot-gate classification, and the probability for the j-th domain-slot pair at the t-th turn is calculated as:

$$P_{tj}^{\text{gate}} = \text{softmax}(\text{FFN}(\mathrm{H}_{L,M+N+j}^G))$$

The loss for slot gate classification is

$$\mathcal{L}_{\text{gate}} = -\sum_{t=1}^{T}\sum_{j=1}^{J}\log(P_{tj}^{\text{gate}} \cdot (y_{tj}^{\text{gate}})^\top)$$

where y_{tj}^{gate} is the one-hot gate label for the j-th domain-slot pair at turn t.

4.3.2 RNN-based Value Decoder

After the slot-gate classification, there are J' domain-slot pairs tagged with PTR class which indicates the domain-slot should take a real value. They are denoted as $\mathbb{C}_t = \{j|\text{argmax}(P_{tj}^{\text{gate}}) = \text{PTR}\}$, and $J' = |\mathbb{C}_t|$.

We use Gated Recurrent Unit (GRU) (Cho et al., 2014) decoder like Wu et al. (2019) and the soft copy mechanism (See et al., 2017) to get the final output distribution $P_{tj}^{\text{value},k}$ over all candidate tokens at the k-th step. More details are illustrated in

Appendix B. The loss function for value decoder is

$$\mathcal{L}_{\text{value}} = -\sum_{t=1}^{T}\sum_{j\in\mathbb{C}_t}\sum_{k}\log(P_{tj}^{\text{value},k}\cdot(y_{tj}^{\text{value},k})^{\top})$$

where $y_{tj}^{\text{value},k}$ is the one-hot token label for the j-th domain-slot pair at k-th step.

During training process, the above modules can be jointly trained and optimized by the summations of different losses as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{gate}} + \mathcal{L}_{\text{value}}$$

5 Experiment

5.1 Datasets

We use MultiWOZ 2.0 (Budzianowski et al., 2018) and MultiWOZ 2.1 (Eric et al., 2019) to evaluate our approach. MultiWOZ 2.0 is a task-oriented dataset of human-human written conversations spanning over seven domains, consists of 10348 multi-turn dialogues. MultiWOZ 2.1 is a revised version of MultiWOZ 2.0, which is re-annotated with a different set of inter-annotators and also canonicalized entity names. According to the work of Eric et al. (2019), about 32% of the state annotations is corrected so that the effect of noise is counteracted.

Note that *hospital* and *police* are excluded since they appear in training set with a very low frequency, and they do not even appear in the test set. To this end, five domains (*restaurant*, *train*, *hotel*, *taxi*, *attraction*) are involved in the experiments with 17 distinct slots and 30 *domain-slot* pairs.

We follow similar data pre-processing procedures as Wu et al. (2019) on both MultiWOZ 2.0 and 2.1. [2] The resulting corpus includes 8,438 multi-turn dialogues in training set with an average of 13.5 turns per dialogue. Data statistics of MultiWOZ 2.1 is shown in Table 1. The adjacency matrix $\mathbf{A}^G$ of MultiWOZ 2.0 and 2.1 datasets is shown in Figure 4 of Appendix, while domain-slot pairs are omitted due to space limitations.

5.2 Experiment Settings

We set the hidden size of CSFN, d_{model}, as 400 with 4 heads. Following Wu et al. (2019), the token embeddings with 400 dimensions are initialized by concatenating Glove embeddings (Pennington et al., 2014) and character embeddings (Hashimoto et al., 2017). We do a grid search over $\{4, 5, 6, 7, 8\}$ for the layer number of CSFN on the validation set. We use a batch size of 32. The DST model is trained using ADAM (Kingma and Ba, 2014) with the learning rate of 1e-4. During training, we use the ground truth of the previous dialogue state and the ground truth value tokens. In the inference, the predicted dialogue state of the last turn is applied, and we use a greedy search strategy in the decoding process of the value decoder.

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

Domain	Slots	*Train*	*Valid*	*Test*
Restaurant	area, food, name, price range, book day, book people, book time	3813	438	437
Hotel	area, internet, name, parking, price range, stars, type, book day, book people, book stay	3381	416	394
Train	arrive by, day, departure, destination, leave at, book people	3103	484	494
Taxi	arrive by, departure, destination, leave at	1654	207	195
Attraction	area, name, type	2717	401	395
Total		8438	1000	1000

Table 1: Data statistics of MultiWOZ2.1.

5.3 Baseline Models

We make a comparison with the following existing models, which are either predefined ontology-based DSTs or open-vocabulary based DSTs. Predefined ontology-based DSTs have the advantage of access to the known candidate set of each slot, while these approaches may not be applicable in the real scenario.

FJST (Eric et al., 2019): It exploits a bidirectional LSTM network to encode the dialog history and a separate FFN to predict the value for each slot.

HJST (Eric et al., 2019): It encodes the dialogue history using an LSTM like FJST, but utilizes a hierarchical network.

SUMBT (Lee et al., 2019): It exploits BERT (Devlin et al., 2018) as the encoder for the dialogue context and slot-value pairs. After that, it scores every candidate slot-value pair with the dialogue context by using a distance measure.

HyST (Goel et al., 2019): It is a hybrid approach based on hierarchical RNNs, which incorporates both a predefined ontology-based setting and an open-vocabulary setting.

DST-Reader (Gao et al., 2019): It models the DST

	Models	BERT used	MultiWOZ 2.0	MultiWOZ 2.1
predefined ontology	HJST (Eric et al., 2019)*	✗	38.40	35.55
	FJST (Eric et al., 2019)*	✗	40.20	38.00
	SUMBT (Lee et al., 2019)	✓	42.40	-
	HyST (Goel et al., 2019)*	✗	42.33	38.10
	DS-DST (Zhang et al., 2019)	✓	-	51.21
	DST-Picklist (Zhang et al., 2019)	✓	-	53.30
	DSTQA (Zhou and Small, 2019)	✗	**51.44**	51.17
	SST (Chen et al., 2020)	✗	51.17	**55.23**
open-vocabulary	DST-Span (Zhang et al., 2019)	✓	-	40.39
	DST-Reader (Gao et al., 2019)*	✗	39.41	36.40
	TRADE (Wu et al., 2019)*	✗	48.60	45.60
	COMER (Ren et al., 2019)	✓	48.79	-
	NADST (Le et al., 2020)	✗	50.52	49.04
	SOM-DST (Kim et al., 2019)	✓	51.72	53.01
	CSFN-DST (ours)	✗	49.59	50.81
	CSFN-DST + BERT (ours)	✓	51.57	52.88
	SOM-DST (our implementation)	✓	51.66	52.85
	SOM-DST + Schema Graph (ours)	✓	**52.23**	**53.19**

Table 2: Joint goal accuracy (%) on the test set of MultiWOZ 2.0 and 2.1. * indicates a result borrowed from Eric et al. (2019). ✓means that a BERT model (Devlin et al., 2018) with contextualized word embeddings is utilized.

from the perspective of text reading comprehensions, and get start and end positions of the corresponding text span in the dialogue context.

DST-Span (Zhang et al., 2019): It treats all domain-slot pairs as span-based slots like DST-Reader, and applies a BERT as the encoder.

DST-Picklist (Zhang et al., 2019): It defines picklist-based slots for classification similarly to SUMBT and applies a pre-trained BERT for the encoder. It relies on a predefined ontology.

DS-DST (Zhang et al., 2019): Similar to HyST, it is a hybrid system of DS-Span and DS-Picklist.

DSTQA (Zhou and Small, 2019): It models multi-domain DST as a question answering problem, and generates a question asking for the value of each domain-slot pair. It heavily relies on a predefined ontology, i.e., the candidate set for each slot is known, except for five time-related slots.

TRADE (Wu et al., 2019): It contains a slot gate module for slots classification and a pointer generator for dialogue state generation.

COMER (Ren et al., 2019): It uses a hierarchical decoder to generate the current dialogue state itself as the target sequence.

NADST (Le et al., 2020): It uses a non-autoregressive decoding scheme to generate the current dialogue state.

SST (Chen et al., 2020): It utilizes a graph attention matching network to fuse information from utterances and schema graphs, and a recurrent graph attention network to control state updating. However, it heavily relies on a predefined ontology.

SOM-DST (Kim et al., 2019): It uses a BERT to jointly encode the previous state, the previous and current dialogue utterances. An RNN-decoder is also applied to generate values for slots that need to be updated in the open-vocabulary setting.

5.4 Main Results

Joint goal accuracy is the evaluation metric in our experiments, which is represented as the ratio of turns whose predicted dialogue states are entirely consistent with the ground truth in the test set. Table 2 illustrates that the joint goal accuracy of CSFN-DST and other baselines on the test set of MultiWOZ 2.0 and MultiWOZ 2.1 datasets.

As shown in the table, our proposed CSFN-DST can outperform other models except for SOM-DST. By combining our schema graphs with SOM-DST, we can achieve state-of-the-art performances on both MultiWOZ 2.0 and 2.1 in the open-vocabulary setting. Additionally, our method using BERT (`Bert-base-uncased`) can obtain very competitive performance with the best systems in the predefined ontology-based setting. When a BERT is exploited, we initialize all parameters of CSFN with the BERT encoder's and initialize the token/position embeddings with the BERT's.

5.5 Analysis

In this subsection, we will conduct some ablation studies to figure out the potential factors for the improvement of our method. (Additional experiments and results are reported in Appendix C, case study is shown in Appendix D.)

Models	Joint Acc. (%)
CSFN-DST	50.81
(-) Omit $\mathrm{H}_L^{B_{t-1}}$ in the decoder	48.66
(-) Omit $\mathrm{H}_L^{X_t}$ in the decoder	48.45
(+) The previous utterance X_{t-1}	50.75

Table 3: Ablation studies for context information on MultiWOZ 2.1.

Models	BERT used	
	✗	✓
CSFN-DST	50.81	52.88
(-) No schema graph, $\mathbf{A}^G = \mathbf{1}$	49.93	52.50
(-) No schema graph, $\mathbf{A}^G = \boldsymbol{I}$	49.52	52.46
(+) Ground truth of the previous state	78.73	80.35
(+) Ground truth slot-gate classifi.	77.31	80.66
(+) Ground truth value generation	56.50	59.12

Table 4: Joint goal accuracy(%) of ablation studies on MultiWOZ 2.1.

5.5.1 Effect of context information

Context information consists of the previous dialogue state or the current dialogue utterance, which are definitely key for the encoder. It would be interesting to know whether the two kinds of context information are also essential for the RNN-based value decoder. As shown in Table 3, we choose to omit the top hidden states of the previous dialogue state ($\mathrm{H}_L^{B_{t-1}}$) or the current utterance ($\mathrm{H}_L^{X_t}$) in the RNN-based value decoder. The results show both of them are crucial for generating real values.

Do we need more context? Only the current dialogue utterance is utilized in our model, which would be more efficient than the previous methods involving all the preceding dialogue utterance. However, we want to ask whether the performance will be improved when more context is used. In Table 3, it shows that incorporating the previous dialogue utterance X_{t-1} gives no improvement, which implies that jointly encoding the current utterance and the previous dialogue state is effective as well as efficient.

5.5.2 Effect of the schema graph

In CSFN-DST, the schema graph with domain-slot relations is exploited. To check the effectiveness of the schema graph used, we remove knowledge-aware domain-slot relations by replacing the adjacency matrix $\mathbf{A}^G$ as a fully connected one $\mathbf{1}$ or node-independent one $\boldsymbol{I}$. Results in Table 4 show that joint goal accuracies of models without the schema graph are decreased similarly when BERT is either used or not.

To reveal why the schema graph with domain-slot relations is essential for joint accuracy, we further make analysis on domain-specific and turn-specific results. As shown in Table 5, the schema graph can benefit almost all domains except for *Attaction (Attr.)*. As illustrated in Table 1, the *Attaction* domain contains only three slots, which should be much simpler than the other domains. Therefore, we may say that the schema graph can help complicated domains.

Models	Attr.	Hotel	Rest.	Taxi	Train
CSFN-DST	64.78	**46.29**	**64.64**	**47.35**	**69.79**
(-) No SG	**65.97**	45.48	62.94	46.42	67.58

Table 5: Domain-specific joint accuracy on MultiWOZ 2.1. SG means Schema Graph.

Turn	Proportion (%)	w/ SG	w/o SG
1	13.6	89.39	88.19 (−1.20)
2	13.6	73.87	72.87 (−1.00)
3	13.4	58.69	57.78 (−0.91)
4	12.8	51.96	50.80 (−1.16)
5	11.9	41.01	39.63 (−1.38)
6	10.7	34.51	35.15 (+0.64)
7	9.1	27.91	29.55 (+1.64)
8	6.3	24.73	23.23 (−1.50)
9	4.0	20.55	19.18 (−1.37)
10	2.3	16.37	12.28 (−4.09)
11	1.3	12.63	8.42 (−4.21)
12	0.6	12.77	8.51 (−4.26)
> 12	0.4	9.09	0.00 (−9.09)
all	100	50.81	49.93

Table 6: Joint accuracies over different dialogue turns on MultiWOZ 2.1. It shows the impact of using schema graph on our proposed CSFN-DST.

The turn-specific results are shown in Table 6, where joint goal accuracies over different dialogue turns are calculated. From the table, we can see that data proportion of larger turn number becomes smaller while the larger turn number refers to more challenging conversation. From the results of the table, we can find the schema graph can make improvements over most dialogue turns.

5.5.3 Oracle experiments

The predicted dialogue state at the last turn is utilized in the inference stage, which is mismatched with the training stage. An oracle experiment is conducted to show the impact of training-inference mismatching, where ground truth of the previous dialogue state is fed into CSFN-DST. The results in Table 4 show that joint accuracy can be nearly 80% with ground truth of the previous dialogue state. Other oracle experiments with ground truth slot-gate classification and ground truth value generation are also conducted, as shown in Table 4.

5.5.4 Slot-gate classification

We conduct experiments to evaluate our model performance on the slot-gate classification task. Table 7 shows F1 scores of the three slot gates, i.e., {NONE, DONTCARE, PTR}. It seems that the pre-trained BERT model helps a lot in detecting slots of which the user doesn't care about values. The F1 score of DONTCARE is much lower than the others', which implies that detecting DONTCARE is a much challenging sub-task.

Gate	CSFN-DST	CSFN-DST + BERT
NONE	99.18	99.19
DONTCARE	72.50	75.96
PTR	97.66	98.05

Table 7: Slot-gate F1 scores on MultiWOZ 2.1.

5.6 Reproducibility

We run our models on GeForce GTX 2080 Ti Graphics Cards, and the average training time for each epoch and number of parameters in each model are provided in Table 8. If BERT is exploited, we accumulate the gradients with 4 steps for a minibatch of data samples (i.e., $32/4 = 8$ samples for each step), due to the limitation of GPU memory. As mentioned in Section 5.4, joint goal accuracy is the evaluation metric used in our experiments, and we follow the computing script provided in TRADE-DST [3].

Method	Time per Batch	# Parameters
CSFN-DST	350ms	63M
CSFN-DST + BERT	840ms	115M
SOM-DST + SG	1160ms	115M

Table 8: Runtime and mode size of our methods.

5.7 Discussion

The main contributions of this work may focus on exploiting the schema graph with graph-based attention networks. Slot-relations are also utilized in DSTQA (Zhou and Small, 2019). However, DSTQA uses a dynamically-evolving knowledge graph for the dialogue context, and we use a static schema graph. We absorb the dialogue context by using the previous (predicted) dialogue state as another input. We believe that the two different usages of the slot relation graph can be complementary. Moreover, these two methods are different in value prediction that DSTQA exploits a hybrid of value classifier and span prediction layer, which relies on a predefined ontology.

SOM-DST (Kim et al., 2019) is very similar to our proposed CSFN-DST with BERT. The main difference between SOM-DST and CSFN-DST is how to exploit the previous dialogue state. For the previous dialogue state, SOM-DST considers all domain-slot pairs and their values (if a domain-slot pair contains an empty value, a special token NONE is used), while CSFN-DST only considers the domain-slot pairs with a non-empty value. Thus, SOM-DST knows which domain-slot pairs are empty and would like to be filled with a value. We think that it is the strength of SOM-DST. However, we choose to omit the domain-slot pairs with an empty value for a lower computation burden, which is proved in Table 8. As shown in the last two rows of Table 2, the schema graph can also improve SOM-DST, which achieves 52.23% and 53.19% joint accuracies on MultiWOZ 2.0 and 2.1, respectively. Appendix E shows how to exploit schema graph in SOM-DST.

6 Conclusion and Future Work

We introduce a multi-domain dialogue state tracker with context and schema fusion networks, which involves slot relations and learns deep representations for each domain-slot pair dependently. Slots from different domains and their relations are organized as a schema graph. Our approach outperforms strong baselines on both MultiWOZ 2.0 and 2.1 benchmarks. Ablation studies also show that the effectiveness of the schema graph.

It will be a future work to incorporate relations among dialogue states, utterances and domain schemata. To further mitigate the data sparsity problem of multi-domain DST, it would be also interesting to incorporate data augmentations (Zhao et al., 2019) and semi-supervised learnings (Lan et al., 2018; Cao et al., 2019).

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments. This work has been supported by Shanghai Jiao Tong University Scientific and Technological Innovation Funds (YG2020YQ01) and No. SKLMCPTS2020003 Project.

[3] https://github.com/jasonwu0731/trade-dst

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 51–64.

Lu Chen, Zhi Chen, Bowen Tan, Sishan Long, Milica Gasic, and Kai Yu. 2019. AgentGraph: Towards universal dialogue management with structured deep reinforcement learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9):1378–1391.

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 *AAAI*, pages 7521–7528.

Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. 2018. Structured dialogue policy with graph neural networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1257–1268.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. MultiWOZ 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. HyST: A hybrid approach for flexible and accurate dialogue state tracking. *arXiv preprint arXiv:1907.00883*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of Annual SIGdial Meeting on Discourse and Dialogue*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of Annual SIGdial Meeting on Discourse and Dialogue*, pages 292–299.

Mettew Henderson, Blaise Thomson, and Jason D. Williams. 2014d. The third dialog state tracking challenge. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2019. Efficient dialogue state tracking by selectively overwriting memory. *arXiv preprint arXiv:1911.03906*.

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

Ouyu Lan, Su Zhu, and Kai Yu. 2018. Semi-supervised training using adversarial multi-task learning for spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6049–6053. IEEE.

Hung Le, Richard Socher, and Steven C.H. Hoi. 2020. Non-autoregressive dialog state tracking. In *International Conference on Learning Representations*.

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.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. *arXiv preprint arXiv:1804.08313*.

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*, pages 1777–1788.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Osman Ramadan, Paweł Budzianowski, and Milica Gasic. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 432–437.

Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1876–1885.

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

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014a. A generalized rule based tracker for dialogue state tracking. In *Proceedings of IEEE Spoken Language Technology Workshop (SLT)*.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014b. The SJTU system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 318–326.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of International Conference on Learning Representations*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of Annual SIGdial Meeting on Discourse and Dialogue*, pages 404–413.

Jason D Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pages 23–24.

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*.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7370–7377.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Kai Yu, Kai Sun, Lu Chen, and Su Zhu. 2015. Constrained markov bayesian polynomial for efficient dialogue state tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12):2177–2188.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*.

Zijian Zhao, Su Zhu, and Kai Yu. 2019. Data augmentation with atomic templates for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3628–3634.

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*, 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*.

A Context- and Schema-Aware Encoding

Besides the hidden states H_i^G of the schema graph G, we show the details of updating $\mathrm{H}_i^{X_t}, \mathrm{H}_i^{B_{t-1}}$ in the i-th layer of CSFN:

$$\mathrm{H}_{i+1}^G, \mathrm{H}_{i+1}^{X_t}, \mathrm{H}_{i+1}^{B_{t-1}} = \text{CSFNLayer}_i(\mathrm{H}_i^G, \mathrm{H}_i^{X_t}, \mathrm{H}_i^{B_{t-1}})$$

The hidden states of the dialogue utterance X_t at the i-the layer are updated as follows:

$$\begin{aligned}
\mathrm{I}_{\mathrm{XX}} &= \text{MultiHead}_{\Theta_{\mathrm{XX}}}(\mathrm{H}_i^{X_t}, \mathrm{H}_i^{X_t}) \\
\mathrm{E}_{\mathrm{XB}} &= \text{MultiHead}_{\Theta_{\mathrm{XB}}}(\mathrm{H}_i^{X_t}, \mathrm{H}_i^{B_{t-1}}) \\
\mathrm{E}_{\mathrm{XG}} &= \text{MultiHead}_{\Theta_{\mathrm{XG}}}(\mathrm{H}_i^{X_t}, \mathrm{H}_i^{G}) \\
\mathrm{C}_X &= \text{LayerNorm}(\mathrm{H}_i^{X_t} + \mathrm{I}_{\mathrm{XX}} + \mathrm{E}_{\mathrm{XB}} + \mathrm{E}_{\mathrm{XG}}) \\
\mathrm{H}_{i+1}^{X_t} &= \text{LayerNorm}(\mathrm{C}_X + \text{FFN}(\mathrm{C}_X))
\end{aligned}$$

where I_{XX} contains internal attention vectors, E_{XB} and E_{XG} are external attention vectors.

The hidden states of the previous dialogue state B_{t-1} at the i-the layer are updated as follows:

$$\begin{aligned}
\mathrm{I}_{\mathrm{BB}} &= \text{GraphMultiHead}_{\Theta_{\mathrm{BB}}}(\mathrm{H}_i^{B_{t-1}}, \mathrm{H}_i^{B_{t-1}}, \mathbf{A}^{B_{t-1}}) \\
\mathrm{E}_{\mathrm{BX}} &= \text{MultiHead}_{\Theta_{\mathrm{BX}}}(\mathrm{H}_i^{B_{t-1}}, \mathrm{H}_i^{X_t}) \\
\mathrm{E}_{\mathrm{BG}} &= \text{MultiHead}_{\Theta_{\mathrm{BG}}}(\mathrm{H}_i^{B_{t-1}}, \mathrm{H}_i^{G}) \\
\mathrm{C}_B &= \text{LayerNorm}(\mathrm{H}_i^{B_{t-1}} + \mathrm{I}_{\mathrm{BB}} + \mathrm{E}_{\mathrm{BX}} + \mathrm{E}_{\mathrm{BG}}) \\
\mathrm{H}_{i+1}^{B_{t-1}} &= \text{LayerNorm}(\mathrm{C}_B + \text{FFN}(\mathrm{C}_B))
\end{aligned}$$

where $\mathbf{A}^{B_{t-1}}$ is the adjacency matrix of the previous dialogue state. The adjacency matrix indicates that each triplets in B_{t-1} is separated, while tokens in a same triplet are connected with each other. The `[CLS]` token is connected with all triplets, serving as a transit node.

B RNN-based Value Decoder

After the slot-gate classification, there are J' domain-slot pairs tagged with `PTR` class which indicates the domain-slot should take a real value. They are denoted as $\mathbb{C}_t = \{j | \text{argmax}(P_{tj}^{\text{gate}}) = \texttt{PTR}\}$, and $J' = |\mathbb{C}_t|$.

We use Gated Recurrent Unit (GRU) (Cho et al., 2014) decoder like Wu et al. (2019) and See et al. (2017). The hidden state $g_{tj}^k \in \mathbb{R}^{1\times d_{\text{model}}}$ is recursively updated by taking a word embedding e_{tj}^k as the input until `[EOS]` token is generated:

$$g_{tj}^k = \text{GRU}(g_{tj}^{k-1}, e_{tj}^k)$$

GRU is initialized with

$$g_{tj}^0 = \mathrm{H}_{L,0}^{X_t} + \mathrm{H}_{L,0}^{B_{t-1}}$$

and $e_{tj}^0 = \mathrm{H}_{L,M+N+j}^G$.

The value generator transforms the hidden state to the probability distribution over the token vocabulary at the k-th step, which consists of two parts: 1) distribution over all input tokens, 2) distribution over the input vocabulary. The first part is computed as

$$P_{tj}^{\text{ctx},k} = \text{softmax}(\textsc{Att}(g_{tj}^k, [\mathrm{H}_L^{X_t}; \mathrm{H}_L^{B_{t-1}}]))$$

where $P_{tj}^{\text{ctx},k} \in \mathbb{R}^{1\times(|X_t|+|B_{t-1}|)}$, and $\textsc{Att}(.,.)$ is a function to get attention weights (Bahdanau et al., 2014) with more details shown in Appendix B.1. The second part is calculated as

$$\begin{aligned}
c_{tj}^k &= P_{tj}^{\text{ctx},k}[\mathrm{H}_L^{X_t}; \mathrm{H}_L^{B_{t-1}}] \\
P_{tj}^{\text{vocab},k} &= \text{softmax}([g_{tj}^k; c_{tj}^k]W_{\text{proj}}E^\top)
\end{aligned}$$

where $P_{tj}^{\text{vocab},k} \in \mathbb{R}^{1\times d_{\text{vocab}}}$, $c_{tj}^k \in \mathbb{R}^{1\times d_{\text{model}}}$ is a context vector, $W_{\text{proj}} \in \mathbb{R}^{2d_{\text{model}}\times d_{\text{model}}}$ is a trainable parameter, and $E \in \mathbb{R}^{d_{\text{vocab}}\times d_{\text{model}}}$ is the token embedding matrix shared across the encoder and the decoder.

We use the soft copy mechanism (See et al., 2017) to get the final output distribution over all candidate tokens:

$$\begin{aligned}
P_{tj}^{\text{value},k} &= p_{\text{gen}}P_{tj}^{\text{vocab},k} + (1-p_{\text{gen}})P_{tj}^{\text{ctx},k} \\
p_{\text{gen}} &= \text{sigmoid}([g_{tj}^k; e_{tj}^k; c_{tj}^k]W_{\text{gen}})
\end{aligned}$$

where $W_{\text{gen}} \in \mathbb{R}^{3d_{\text{model}}\times 1}$ is a trainable parameter.

The loss function for value decoder is

$$\mathcal{L}_{\text{value}} = -\sum_{t=1}^{T}\sum_{j\in\mathbb{C}_t}\sum_k \log(P_{tj}^{\text{value},k}\cdot(y_{tj}^{\text{value},k})^\top)$$

where $y_{tj}^{\text{value},k}$ is the one-hot token label for the j-th domain-slot pair at k-th step.

B.1 Attention Weights

For attention mechanism for computing $P_{tj}^{\text{ctx},k}$ in the RNN-based value decoder, we follow Bahdanau et al. (2014) and define the $\textsc{Att}(.,.)$ function as

$$\begin{aligned}
u_i &= \tanh(\mathbf{x}\mathbf{W}_1^{\text{att}} + \mathbf{h}_i\mathbf{W}_2^{\text{att}} + \mathbf{b}^{\text{att}})\mathbf{v}^\top \\
a_i &= \frac{\exp(u_i)}{\sum_{j=1}^{S}\exp(u_j)} \\
\mathbf{a} &= \{a_1, \cdots, a_S\} = \textsc{Att}(\mathbf{x}, \mathrm{H})
\end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^{1\times d}$, $\mathrm{H} \in \mathbb{R}^{S\times d}$, $\mathbf{W}_1^{\text{att}} \in \mathbb{R}^{d\times d}$, $\mathbf{W}_2^{\text{att}} \in \mathbb{R}^{d\times d}$, $\mathbf{b}^{\text{att}} \in \mathbb{R}^{1\times d}$, $\mathbf{v} \in \mathbb{R}^{1\times d}$, and $\mathbf{h}_i$ is the i-th row vector of H. Therefore, $\textsc{Att}(\mathbf{x}, \mathrm{H})$ returns an attention distribution of $\mathbf{x}$ over H.

C Additional Results

Domain-specific Results Domain-specific accuracy is the joint goal accuracy measured on a subset of the predicted dialogue state, which only contains the slots belong to a domain. From the results of Table 9, we can find BERT can make improvements on all domains, and especially the improvement on *Taxi* domain is the largest.

Domain	CSFN-DST	CSFN-DST + BERT
Attraction	64.78	67.82
Hotel	46.29	48.80
Restaurant	64.64	65.23
Taxi	47.35	53.58
Train	69.79	70.91

Table 9: Domain-specific joint accuracy on MultiWOZ 2.1.

Slot-specific Results Slot-specific F1 score is measured for predicting slot-value pairs of the corresponding slot. Table 10 shows slot-specific F1 scores of CSFN-DST without the schema graph, CSFN-DST and CSFN-DST with BERT on the test set of MultiWOZ 2.1.

D Case Study

We also conduct case study on the test set of MultiWOZ 2.1, and four cases are shown in Table 11. From the first three cases, we can see the schema graph can copy values from related slots in the memory (i.e., the previous dialogue state). In the case C1, the model makes the accurate reference of the phrase "*whole group*" through the context, and the value of *restaurant-book_people* is copied as the value of *train-book_people*. We can also see a failed case (C4). It is too complicated to inference the *departure* and *destination* by a word "*commute*".

E SOM-DST with Schema Graph

For SOM-DST (Kim et al., 2019), the input tokens to the state operation predictor are the concatenation of the previous turn dialog utterances, the current turn dialog utterances, and the previous turn dialog state:

$$X_t = [\texttt{CLS}] \oplus D_{t-1} \oplus D_t \oplus B_{t-1},$$

where D_{t-1} and D_t are the last and current utterances, respectively. The dialogue state B_t is denoted as $B_t = B_t^1 \oplus \ldots \oplus B_t^J$, where $B_t^j = [\texttt{SLOT}]^j \oplus S^j \oplus - \oplus V_t^j$ is the representation of the j-th slot-value pair. To incorporate the schema graph, we exploit the special token $[\texttt{SLOT}]^j$ to replace the domain-slot node o_j in the schema graph ($j = 1, \cdots, J$). Then, domain and slot nodes $G' = d_1 \oplus \cdots \oplus d_M \oplus s_1 \oplus \cdots \oplus s_N$ are concatenated into X_t, i.e.,

$$X_t = [\texttt{CLS}] \oplus D_{t-1} \oplus D_t \oplus B_{t-1} \oplus G',$$

where the relations among domain, slot and domain-slot nodes are also considered in attention masks of BERT.

	attraction	hotel	restaurant	taxi	train	area	name	type	book_day	book_people	book_stay	internet	parking	price_range	stars	book_time	food	arrive_by	departure	destination	leave_at	day
attraction	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
hotel	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0
restaurant	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	0	0	0	**1**	0	**1**	**1**	0	0	0	0	0
taxi	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	0
train	**1**	**1**	**1**	**1**	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	**1**
area	**1**	**1**	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
name	**1**	**1**	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0
type	**1**	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
book_day	0	**1**	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**
book_people	0	**1**	**1**	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
book_stay	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
internet	0	**1**	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
parking	0	**1**	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
price_range	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
stars	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
book_time	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
food	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
arrive_by	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	0
departure	0	0	0	**1**	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0
destination	0	0	0	**1**	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0
leave_at	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	0
day	0	0	0	0	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**

Figure 4: Adjacency matrix $\mathbf{A}^G$ of MultiWOZ 2.0 and 2.1 datasets. It contains only domain and slot nodes, while domain-slot paris are omitted due to space limitations. The first five items are domains ("attraction, hotel, restaurant, taxi, train"), and the rest are slots.

Domain-slot	CSFN-DST (no SG)	CSFN-DST	CSFN-DST + BERT
attraction-area	91.67	91.92	**92.81**
attraction-name	78.28	77.77	**79.55**
attraction-type	90.95	90.89	**91.97**
hotel-area	84.59	84.21	**84.86**
hotel-book_day	97.16	**97.79**	97.03
hotel-book_people	95.43	96.14	**97.35**
hotel-book_stay	96.04	**97.00**	96.98
hotel-internet	86.79	**89.98**	86.58
hotel-name	82.97	83.11	**84.61**
hotel-parking	86.68	**87.66**	86.07
hotel-price_range	89.09	90.10	**92.56**
hotel-stars	91.51	**93.49**	93.34
hotel-type	77.58	77.87	**82.12**
restaurant-area	93.73	**94.27**	94.09
restaurant-book_day	**97.75**	97.66	97.42
restaurant-book_people	96.79	96.67	**97.84**
restaurant-book_time	92.43	91.60	**94.29**
restaurant-food	**94.90**	94.18	94.48
restaurant-name	80.72	**81.39**	80.59
restaurant-price_range	93.47	94.28	**94.49**
taxi-arrive_by	78.81	81.09	**86.08**
taxi-departure	73.15	71.39	**75.15**
taxi-destination	73.79	78.06	**79.83**
taxi-leave_at	77.29	80.13	**88.06**
train-arrive_by	86.43	87.56	**88.77**
train-book_people	89.59	91.41	**92.33**
train-day	98.44	98.41	**98.52**
train-departure	95.91	**96.52**	96.22
train-destination	**97.08**	97.06	96.13
train-leave_at	69.97	70.97	**74.50**
Joint Acc. overall	49.93	50.81	52.88

Table 10: Slot-specific F1 scores on MultiWOZ 2.1. SG means Schema Graph. The results in bold black are the best slot F1 scores.

C1	**Previous DS:**	(restaurant-book_day, friday), (restaurant-book_people, **8**), (restaurant-book_time, 10:15), (restaurant-name, restaurant 2 two), (train-leave_at, 12:15), (train-destination, peterborough), (train-day, saturday), (train-departure, cambridge)
	System:	How about train tr3934? It leaves at 12:34 & arrives at 13:24. Travel time is 50 minutes.
	Human:	That sounds fine. Can I get tickets for my **whole group** please?
	Gold DS:	(restaurant-name, restaurant 2 two), (restaurant-book_day, friday), (restaurant-book_people, 8), (restaurant-book_time, 10:15), (train-departure, cambridge), (train-leave_at, 12:15), (train-day, saturday), (train-destination, peterborough), (train-book_people, **8**)
	CSFN-DST (no SG):	(restaurant-name, restaurant 2 two), (restaurant-book_day, friday), (restaurant-book_people, 8), (restaurant-book_time, 10:15), (train-departure, cambridge), (train-leave_at, 12:15), (train-day, saturday), (train-destination, peterborough), (train-book_people, **1**)
	CSFN-DST:	(restaurant-name, restaurant 2 two), (restaurant-book_day, friday), (restaurant-book_people, 8), (restaurant-book_time, 10:15), (train-departure, cambridge), (train-leave_at, 12:15), (train-day, saturday), (train-destination, peterborough), (train-book_people, **8**)
C2	**Previous DS:**	(hotel-area, west), (hotel-price_range, cheap), (hotel-type, guest house), (hotel-internet, yes), (hotel-name, warkworth house), (restaurant-area, centre), (restaurant-food, italian), (restaurant-price_range, cheap), (restaurant-name, **ask**)
	System:	01223364917 is the phone number. 12 bridge street city centre, cb21uf is the address.
	Human:	Thanks. I will also need a taxi from **the hotel** to **the restaurant**. Will you handle this?
	Gold DS:	(hotel-area, west), (hotel-price_range, cheap), (hotel-type, guest house), (hotel-internet, yes), (hotel-name, warkworth house), (restaurant-area, centre), (restaurant-food, italian), (restaurant-price_range: cheap), (restaurant-name, ask), (taxi-departure, warkworth house), (taxi-destination, **ask**)
	CSFN-DST (no SG):	(hotel-area, west), (hotel-price_range, cheap), (hotel-type, guest house), (hotel-internet, yes), (hotel-name, warkworth house), (restaurant-area, centre), (restaurant-food, italian), (restaurant-price_range: cheap), (restaurant-name, ask), (taxi-departure, warkworth house), (taxi-destination, **warkworth house**)
	CSFN-DST:	(hotel-area, west), (hotel-price_range, cheap), (hotel-type, guest house), (hotel-internet, yes), (hotel-name, warkworth house), (restaurant-area, centre), (restaurant-food, italian), (restaurant-price_range: cheap), (restaurant-name, ask), (taxi-departure, warkworth house), (taxi-destination, **ask**)
C3	**Previous DS:**	(attraction-area, east), (attraction-name, funky fun house), (restaurant-area, east), (restaurant-food, indian), (restaurant-price_range, moderate), (restaurant-name, **curry prince**)
	System:	cb58jj is there postcode. Their address is 451 newmarket road fen ditton.
	Human:	Great, thank you! Also, can you please book me a taxi between the restaurant and funky fun house? I want to leave **the restaurant** by 01:30.
	Gold DS:	(attraction-area, east), (attraction-name, funky fun house), (restaurant-area, east), (restaurant-food, indian), (restaurant-price_range, moderate), (restaurant-name, curry prince), (taxi-departure, **curry prince**), (taxi-destination, funky fun house), (taxi-leave_at, 01:30)
	CSFN-DST (no SG):	(attraction-area, east), (attraction-name, funky fun house), (restaurant-area, east), (restaurant-food, indian), (restaurant-price_range, moderate), (restaurant-name, curry prince), (taxi-departure, **curry garden**), (taxi-destination, funky fun house), (taxi-leave_at, 01:30)
	CSFN-DST:	(attraction-area, east), (attraction-name, funky fun house), (restaurant-area, east), (restaurant-food, indian), (restaurant-price_range, moderate), (restaurant-name, curry prince), (taxi-departure, **curry prince**), (taxi-destination, funky fun house), (taxi-leave_at, 01:30)
C4	**Previous DS:**	(hotel-name, **a and b guest house**), (hotel-book_day, tuesday), (hotel-book_people, 6), (hotel-book_stay, 4), (attraction-area, west), (attraction-type, museum)
	System:	**Cafe jello gallery** has a free entrance fee. The address is cafe jello gallery, 13 magdalene street and the post code is cb30af. Can I help you with anything else?
	Human:	Yes please. I need a taxi to **commute**.
	Gold DS:	(hotel-name, a and b guest house), (hotel-book_day, tuesday), (hotel-book_people, 6), (hotel-book_stay, 4), (attraction-area, west), (attraction-type, museum), (taxi-destination, **cafe jello gallery**), (taxi-departure, **a and b guest house**)
	CSFN-DST (no SG):	(hotel-name, a and b guest house), (hotel-book_day, tuesday), (hotel-book_people, 6), (hotel-book_stay, 4), (attraction-area, west), (attraction-type, museum)
	CSFN-DST:	(hotel-name, a and b guest house), (hotel-book_day, tuesday), (hotel-book_people, 6), (hotel-book_stay, 4), (attraction-area, west), (attraction-type, museum), (taxi-destination, **cafe jello gallery**)

Table 11: Four cases on the test set of MultiWOZ 2.1. DS means Dialogue State, and SG means Schema Graph.