UniConv: A Unified Conversational Neural Architecture for Multi-domain Task-oriented Dialogues

Hung Le[†§], **Doyen Sahoo**[‡], **Chenghao Liu**[†], **Nancy F. Chen**[§], **Steven C.H. Hoi**[†‡]
[†] Singapore Management University
{hungle.2018,chliu}@smu.edu.sg
[‡] Salesforce Research Asia
{dsahoo,shoi}@salesforce.com
[§]Institute for Infocomm Research, A*STAR
nfychen@i2r.a-star.edu.sg

Abstract

Building an end-to-end conversational agent for multi-domain task-oriented dialogue has been an open challenge for two main reasons. First, tracking dialogue states of multiple domains is non-trivial as the dialogue agent must obtain complete states from all relevant domains, some of which might have shared slots among domains as well as unique slots specifically for one domain only. Second, the dialogue agent must also process various types of information across domains, including dialogue context, dialogue states, and database, to generate natural responses to users. Unlike the existing approaches that are often designed to train each module separately, we propose "UniConv" — a novel unified neural architecture for end-to-end conversational systems in multi-domain task-oriented dialogues, which is designed to jointly train (i) a Bi-level State Tracker which tracks dialogue states by learning signals at both slot and domain level independently, and (ii) a Joint Dialogue Act and Response Generator which incorporates information from various input components and models dialogue acts and target responses simultaneously. We conduct comprehensive experiments in dialogue state tracking, context-to-text, and end-to-end settings on the MultiWOZ2.1 benchmark, achieving superior performance over competitive baselines in all tasks. Our code and models will be released.

1 Introduction

A conventional approach to task-oriented dialogues is to solve four distinct tasks: (1) natural language understanding (NLU) which parses user utterance into a semantic frame, (2) dialogue state tracking (DST) which updates the slots and values from semantic frames to the latest values for knowledge base retrieval, (3) dialogue policy which determines an appropriate dialogue act for the next system response, and (4) response generation which generates a natural language sequence conditioned on the dialogue act. This traditional pipeline modular framework has achieved remarkable successes in task-oriented dialogues (Wen et al., 2017; Liu and Lane, 2017; Williams et al., 2017; Zhao et al., 2017). However, such kind of dialogue system is not fully optimized as the modules are loosely integrated and often not trained jointly in an end-to-end manner, and thus may suffer from increasing error propagation between the modules as the complexity of the dialogues evolves.

A typical case of a complex dialogue setting is when the dialogue extends over multiple domains. A dialogue state in a multi-domain dialogue should include slots of all applicable domains up to the current turn (See Table 1). Each domain can have shared slots that are common among domains or unique slots that are not shared with any. Directly applying single-domain DST to multi-domain dialogues is not straightforward because the dialogue states extend to multiple domains. A possible approach is to process a dialogue of N_D domains multiple times, each time obtaining a dialogue state of one domain. However, this approach does not allow learning co-reference in dialogues in which users can switch from one domain to another.

As the number of dialogue domains increases, traditional pipeline approaches propagate errors from dialogue states to dialogue policy and subsequently, to natural language generator. Recent efforts (Eric et al., 2017; Madotto et al., 2018; Wu et al., 2019b) address this problem with an integrated sequence-to-sequence structure. These approaches often consider knowledge bases as memory tuples rather than relational entity tables. While achieving impressive performance, these approaches are not scalable to large-scale knowledge-bases, e.g. thousands of entities, as the memory cost to query entity attributes increases substantially. Another limitation in these approaches is the

Human: hello *Dialogue state: {}*	**Dialogue agent**: hi ! how can i help you ? *Dialogue acts: [greeting]*
Human: i would like a place to eat in the expensive price range *Dialogue state: {restaurant: {pricerange: expensive}}*	**Dialogue agent**: sure , what type of food are you interested in ? *Dialogue acts: [request-food]*
Human: could you make a suggestion ? one in the centre ? *Dialogue state: {restaurant: {pricerange: expensive, area: centre}}* ...	**Dialogue agent**: fitzbillies restaurant is an expensive british restaurant in the centre . can i book that for you ? *Dialogue acts: [inform-restaurant, request-booking]* ...
Human: also , i need the number for kings hedges learner pool . *Dialogue state: {restaurant: {pricerange: expensive, area: centre, name=fizbillies restaurant, request=[address]}, attraction: {name: kings hedges learner pool, request=[phone]}}*	**Dialogue agent**: the phone number for the pool is 01234567 , is there something else i can help you ? *Dialogue acts: [inform-phone]*

Table 1: Example of a multi-domain dialogue with two domains: *restaurant* and *attraction*. Each row represents a dialogue turn with annotation of dialogue state and dialogue acts.

absence of dialogue act modelling. Dialogue act is particularly important in task-oriented dialogues as it determines the general decision towards task completion before a dialogue agent can materialize it into natural language response (See Table 1).

To tackle the challenges in multi-domain task-oriented dialogues while reducing error propagation among dialogue system modules and keeping the models scalable, we propose UniConv, a unified neural network architecture for end-to-end dialogue systems. UniConv consists of a Bi-level State Tracking (BDST) module which embeds natural language understanding as it can directly parse dialogue context into a structured dialogue state rather than relying on the semantic frame output from an NLU module in each dialogue turn. BDST implicit models and integrates slot representations from dialogue contextual cues to directly generate slot values in each turn and thus, remove the need for explicit slot tagging features from an NLU. This approach is more practical than the traditional pipeline models as we do not need slot tagging annotation. Furthermore, BDST tracks dialogue states in dialogue context in both slot and domain levels. The output representations from two levels are combined in a *late fusion* approach to learn multi-domain dialogue states. Our dialogue state tracker disentangles slot and domain representation learning while enabling deep learning of shared representations of slots common among domains.

UniConv integrates BDST with a Joint Dialogue Act and Response Generator (DARG) that simultaneously models both dialogue acts and generates system responses by learning a latent variable representing dialogue acts and semantically conditioning output response sequences on this latent variable. The multi-task setting of DARG allows our models to model dialogue acts while utilizing the distributed representations of dialogue acts, rather than hard discrete output values from a dialogue policy module, on output response tokens. Our response generator incorporates information from dialogue input components and intermediate representations progressively over multiple attention steps. The output representations are refined after each step to obtain high-resolution signals needed to generate appropriate dialogue acts and responses. We combine both BDST and DARG for end-to-end neural dialogue systems, from input dialogues to output system responses.

We evaluate our models on the large-scale MultiWOZ benchmark (Budzianowski et al., 2018), and compare with the existing methods in DST, context-to-text generation, and end-to-end settings. The promising performance in all tasks validates the efficacy of our method.

2 Related Work

2.1 Dialogue State Tracking

Traditionally, DST models are designed to track states of single-domain dialogues such as WOZ (Wen et al., 2017) and DSTC2 (Henderson et al., 2014a) benchmarks. There have been recent efforts that aim to tackle multi-domain DST such as (Ramadan et al., 2018; Lee et al., 2019; Wu et al., 2019a; Goel et al., 2019). These models can be categorized into two main categories: Fixed vocabulary models (Zhong et al., 2018; Ramadan et al., 2018; Lee et al., 2019), which assume known slot ontology with a fixed candidate set for each slot. On the other hand, open-vocabulary models (Lei et al., 2018; Wu et al., 2019a; Gao et al., 2019; Ren et al., 2019; Le et al., 2020) derive the candidate set based on the source sequence i.e. dialogue history, itself. Our approach is more related to the open-vocabulary approach as we aim to generate unique dialogue states depending on the input di-

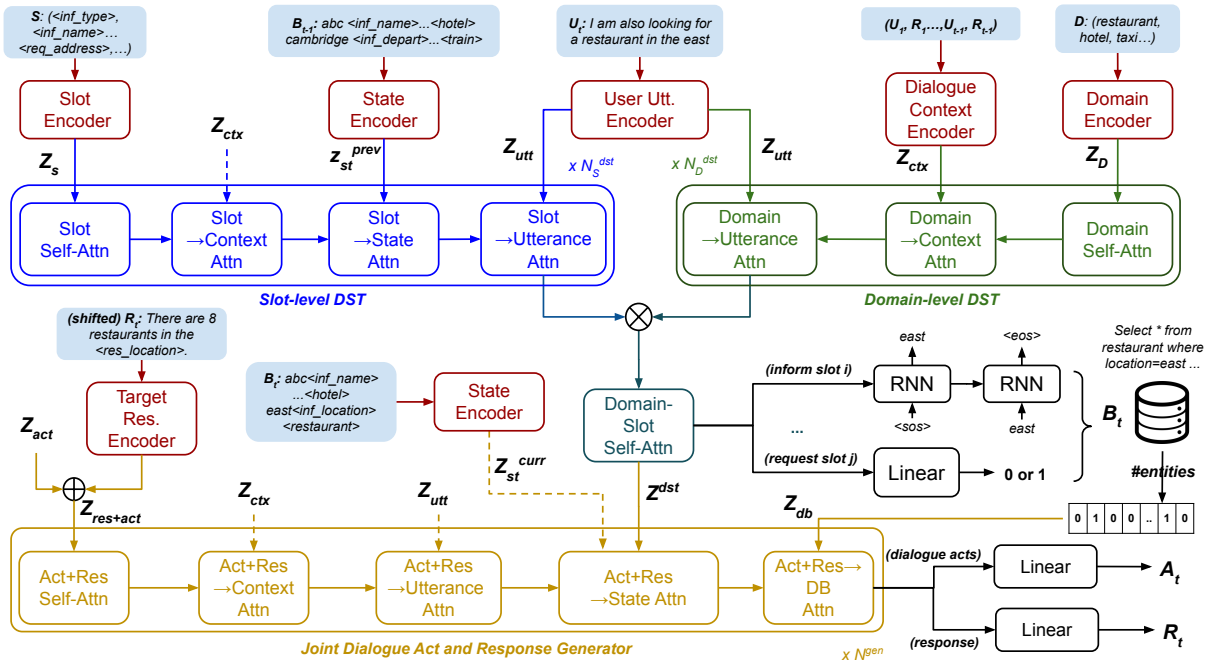

Figure 1: Our unified architecture has three components: (1) *Encoders* encode all text input into continuous representations. (2) *Bi-level State Tracker (BDST)* is used to detect contextual dependencies to generate dialogue states. The DST includes 2 modules for slot-level and domain-level representation learning. (3) *Joint Dialogue Act and Response Generator (DARG)* obtains dependencies between the target response representations and other dialogue components.

alogue. Different from previous generation-based approaches, our state tracker can incorporate contextual information into domain and slot representations and explicitly learns dependencies among slots and domains independently.

2.2 Context-to-Text Generation

This task was traditionally solved by two separate dialogue modules: Dialogue Policy (Peng et al., 2017, 2018) and NLG (Wen et al., 2016; Su et al., 2018). Recent work attempts to combine these two modules to directly generate system responses with or without modeling dialogue acts. Zhao et al. (2019) models action space of dialogue agent as latent variables. Chen et al. (2019) predicts dialogue acts using a hierarchical graph structure with each path representing a unique act. Pei et al. (2019); Peng et al. (2019) use multiple dialogue agents, each trained for a specific dialogue domain, and combine them through a main dialogue agent. Mehri et al. (2019) models dialogue policy and NLG separately and fuses feature representations at different levels to generate responses. Our models learn dialogue acts as a latent variable through a multi-labeled classification task to allow multiple dialogue acts in each turn while semantically conditioning all target tokens on this latent variable in each generation step. This approach emphasizes the importance of dialogue acts in generating dialogue responses while allowing semantic conditioning on distributed representations of dialogue acts rather than hard discrete features.

2.3 End-to-End Dialogue Systems

In this task, conventional approaches combine Natural Language Understanding (NLU), DST, Dialogue Policy, and NLG, into a pipeline architecture (Wen et al., 2017; Bordes et al., 2016; Liu and Lane, 2017; Li et al., 2017; Liu and Perez, 2017; Williams et al., 2017; Zhao et al., 2017). Another framework does not explicitly modularize these components but incorporate them through a sequence-to-sequence framework (Serban et al., 2016; Lei et al., 2018; Yavuz et al., 2019) and a memory-based entity dataset of triplets (Eric and Manning, 2017; Eric et al., 2017; Madotto et al., 2018; Qin et al., 2019; Gangi Reddy et al., 2019; Wu et al., 2019b). These approaches bypass dialogue state and/or act modeling and aim to generate output responses directly. They achieve impressive success in generating dialogue responses in open-domain dialogues with unstructured knowledge bases. However, in a task-oriented setting with an entity dataset, they might suffer from an explosion of memory size when the number of entities increases, especially when there are multiple

datasets from multiple dialogue domains. Our work is more related to the traditional pipeline strategy as we allow our model to explicitly learn dialogue states and acts and enable efficient database search. We integrate our dialogue models by unifying two major components rather than using the traditional four-module architecture, to alleviate error propagation from upstream to downstream components. Similar to our work, Shu et al. (2019) proposes a novel end-to-end architecture for task-oriented dialogue systems. Different from this work, our model facilitates multi-domain state tracking and allows learning dialogue acts during response generation.

3 Method

The input consists of dialogue context of $t-1$ turns, each including a pair of user utterance U and system response R, $(U_1, R_1), ..., (U_{t-1}, R_{t-1})$, and the user utterance at current turn U_t. A task-oriented dialogue system aims to generate the next response R_t, that is not only appropriate to the dialogue context, but also contains the correct information relevant to the current dialogue domains. The information is typically queried from a database based on the user's provided information i.e. *inform* slots tracked by a DST. We assume access to a database of all domains with each column corresponding to a specific slot being tracked. We denote the intermediate output, including the dialogue state of current turn B_t and dialogue act as A_t. We denote the list of all domains $D = (d_1, d_2, ...)$, all slots $S = (s_1, s_2, ...)$, and all acts $A = (a_1, a_2, ...)$. We also denote the list of all (domain, slot) pairs as $DS = (ds_1, ds_2, ...)$. Note that $\|DS\| \leq \|D\| \times \|S\|$ as some slots might not be applicable in all domains. Without loss of generalization, given the current dialogue turn t, we represent each text input as a sequence of tokens, each of which is a unique token index from a vocabulary set V: dialogue context X_{ctx}, current user utterance X_{utt}, and target system response X_{res}. Similarly, we also represent the list of domains as X_D and the list of slots as X_S.

In DST, following similar approaches in (Lei et al., 2018; Budzianowski and Vulić, 2019), we consider the raw text form of dialogue state of the previous turn B_{t-1} using the following template:

$\langle\text{value1}\rangle\langle\text{slot1}\rangle\langle\text{value2}\rangle\langle\text{slot2}\rangle...\langle\text{domain1}\rangle$
$\langle\text{value1}\rangle\langle\text{slot1}\rangle\langle\text{value2}\rangle\langle\text{slot2}\rangle...\langle\text{domain2}\rangle...$

In the context-to-text setting which assumes access to the ground-truth dialogue states of current turn B_t, we also consider the raw text form using the same template. The dialogue state of the previous and current turn can then be represented as a sequence of tokens $X_{\text{st}}^{\text{prev}}$ and $X_{\text{st}}^{\text{curr}}$ respectively. For a fair comparison with current approaches, during inference, we use the model predicted dialogue states $\hat{X}_{\text{st}}^{\text{prev}}$ and do not use $X_{\text{st}}^{\text{curr}}$ in DST and end-to-end tasks. For X_{res}, following current approaches of response generation (Wen et al., 2015; Budzianowski et al., 2018), we consider the delexicalized target response as input by replacing tokens of slot values by their corresponding generic tokens to allow learning value-independent parameters. We denote the delexicalized response $X_{\text{res}}^{\text{dl}}$.

Our model consists of 3 major components (See Figure 1). First, *Encoders* encode all text input into continuous representations. To make it consistent, we encode all input with the same embedding dimension. Secondly, our *Bi-level State Tracker (BDST)* is used to detect contextual dependencies to generate dialogue states. The DST includes 2 modules for slot-level and domain-level representation learning. Each module comprises attention layers to project domain or slot representations and incorporate important information from dialogue context, dialogue state of the previous turn, and current user utterance. The outputs of the two modules are combined to create domain-slot joint feature representations. They are used as a context-aware vector to decode the corresponding *inform* or *request* slots in each domain. Lastly, our *Joint Dialogue Act and Response Generator (DARG)* projects the target system response representations and enhances them with information from various dialogue components. Our response generator can also learn a latent representation to generate dialogue acts, which condition all target tokens during each generation step.

3.1 Encoders

An encoder encodes a text sequence X to a sequence of continuous representation $Z \in \mathbb{R}^{L_X \times d}$. L_X is the length of sequence X and d is the embedding dimension. Each encoder includes a token-level embedding layer. The embedding layer is a trainable embedding matrix $E \in \mathbb{R}^{\|V\| \times d}$. Each row represents a token in the vocabulary set V as a d-dimensional vector. We denote $E(X)$ as the embedding function that transform the sequence X by looking up the respective token index: $Z_{\text{emb}} = E(X) \in \mathbb{R}^{L_X \times d}$. We inject the posi-

tional attribute of each token as similarly adopted in (Vaswani et al., 2017). The positional encoding is denoted as PE. The final embedding is the element-wise summation between token-embedded representations and positional encoded representations with layer normalization (Ba et al., 2016).

$$Z = \text{LayerNorm}(Z_{\text{emb}} + PE(X)) \in \mathbb{R}^{L_X \times d}$$

The encoder outputs include representations of dialogue context Z_{ctx}, current user utterance Z_{utt}, and target response $Z^{\text{dl}}_{\text{res}}$. We also encode the dialogue states of the previous turn and current turn and obtain $Z^{\text{prev}}_{\text{st}}$ and $Z^{\text{curr}}_{\text{st}}$ respectively. We encode X_S and X_D using only token-level embedding layer: $Z_S = \text{LayerNorm}(E(X_S))$ and $Z_D = \text{LayerNorm}(E(X_D))$. During training, we shift the target response by one position to the left side to allow auto-regressive prediction in each generation step. We share the embedding matrix E to encode all text tokens except for tokens of target responses. We use a separate embedding matrix E_{res} to encode target tokens as the delexicalized outputs contain different semantic dynamics from the original source sequences.

3.2 Bi-level State Tracker

3.2.1 Slot-level DST

We use a transformer-based neural network (Vaswani et al., 2017), consisting of dot-product attention from one representation to another, together with skip connection, to integrate dialogue contextual information into each slot representations. We denote $\text{Att}(Z_1, Z_2)$ for attention from Z_2 on Z_1:

$$Z^{(1)} = Z_1 W^{(1)} \in \mathbb{R}^{L_{Z_1} \times d_{\text{att}}} \quad (1)$$

$$Z^{(2)} = Z_2 W^{(2)} \in \mathbb{R}^{L_{Z_2} \times d_{\text{att}}} \quad (2)$$

$$S = \text{Softmax}(Z^{(2)} Z^{(1)}) \in \mathbb{R}^{L_{Z_2} \times L_{Z_1}} \quad (3)$$

$$Z^{(3)} = \text{ReLU}((SZ_1)W^{(3)}) \in \mathbb{R}^{L_{Z_2} \times d} \quad (4)$$

$$Z^{(4)} = \text{LayerNorm}(Z_2 + Z^{(3)}) \in \mathbb{R}^{L_{Z_2} \times d} \quad (5)$$

where $W^{(1)}, W^{(2)}, W^{(3)}$ each has dimensions $\mathbb{R}^{d \times d_{\text{att}}}$. The attention mechanism can be enhanced with multi-head structure in which h_{att} linear transformation layers are used to project each input representation to different feature spaces. The output representations of equation 5 from all heads are then concatenated as the final output.

Slot Self-Attention. We first enable models to process all slot representations together rather than separately as in previous DST models (Ramadan et al., 2018; Wu et al., 2019a). This strategy allows our models to explicitly learn dependencies between all pairs of slots. Many pairs of slots could exhibit correlation such as time-wise relation ("departure_time" and "arrival_time"). We obtain $Z^{\text{dst}}_{SS} = \text{Att}(Z_S, Z_S) \in \mathbb{R}^{\|S\| \times d}$.

Slot→Dialogue Attention. We incorporate the dialogue information by learning dependencies between each slot representation and each token in the dialogue history. Traditional approaches consider all dialogue history as a single sequence, i.e. combining both X_{ctx} and X_{utt}. However, we separate them into two inputs because the information in X_{utt} is usually more important to generate responses while X_{ctx} includes more background information. We then obtain $Z^{\text{dst}}_{S,\text{ctx}} = \text{Att}(Z_{\text{ctx}}, Z^{\text{dst}}_{SS}) \in \mathbb{R}^{\|S\| \times d}$ and $Z^{\text{dst}}_{S,\text{utt}} = \text{Att}(Z_{\text{utt}}, Z^{\text{dst}}_{\text{S,ctx}}) \in \mathbb{R}^{\|S\| \times d}$ sequentially. Following (Lei et al., 2018), we incorporate dialogue state of the previous turn B_{t-1} which is a more compact representation of dialogue context. Hence, we can replace the full dialogue context to only R_{t-1} as the remaining part is represented in B_{t-1}. This approach avoids taking in all dialogue history and is scalable as the conversation grows longer. We then add the attention layer to obtain $Z^{\text{dst}}_{S,\text{st}} = \text{Att}(Z^{\text{prev}}_{\text{st}}, Z^{\text{dst}}_{S,\text{ctx}}) \in \mathbb{R}^{\|S\| \times d}$ (See Figure 1). Using dialogue state of the previous turn provides a more information-intensive input yet requires less memory than processing a full-length dialogue context input. We further improve the feature representations by learning over N^{dst}_S times. At the end of each round, the representation $Z^{\text{dst}}_{S,\text{utt}}$ is used as Z_2 in equations 1 to 5 in the next attention block. We denote the final output Z^{dst}_S.

3.2.2 Domain-level DST

We adopt a similar architecture to learn domain-level representations. We can consider the representations learned in this module exhibiting global relationships while slot-level representations containing local dependencies.

Domain Self-Attention. First, our DST can capture dependencies between all pairs of domains. For example, some domains such as "hotel", "attraction", and "taxi" are usually combined in a dialogue episode of travel planning. We then obtain $Z^{\text{dst}}_{DD} = \text{Att}(Z_D, Z_D) \in \mathbb{R}^{\|D\| \times d}$.

Domain→Dialogue Attention. We allow models to capture dependencies between each domain representation and each token in dialogue context and current user utterance. By segregating dia-

logue context and current utterance, our models can potentially detect changes of dialogue domains from past turns to the current turn. Especially in multi-domain dialogues, users can switch from one domain to another and the next system response should address the latest domain. We then obtain $Z^{\text{dst}}_{D,\text{ctx}} = \text{Att}(Z_{\text{ctx}}, Z^{\text{dst}}_{DD}) \in \mathbb{R}^{\|D\|\times d}$ and $Z^{\text{dst}}_{D,\text{utt}} = \text{Att}(Z_{\text{utt}}, Z^{\text{dst}}_{D,\text{ctx}}) \in \mathbb{R}^{\|D\|\times d}$ sequentially. Different from slot-level DST, we do not use dialogue state of the previous turn as input because we expect global dependencies in domain representations are easier to detect. Similar to the slot-level module, we refine feature representations over N^{dst}_D times and denote the final output as Z^{dst}_D.

3.2.3 Domain-Slot DST

We combined domain and slot representations by expanding the tensors to identical dimensions i.e. $\|D\| \times \|S\| \times d$. We then apply Hadamard product, resulting in domain-slot joint features $Z^{\text{dst}}_{DS} \in \mathbb{R}^{\|D\|\times\|S\|\times d}$. We then apply a self-attention layer to allow learning of dependencies between joint domain-slot features: $Z^{\text{dst}} = \text{Att}(Z^{\text{dst}}_{DS}, Z^{\text{dst}}_{DS}) \in \mathbb{R}^{\|D\|\times\|S\|\times d}$. In this attention, we mask the intermediate representations in positions of invalid domain-slot pairs. Compared to previous work such as (Wu et al., 2019a), we adopt a *late fusion* method whereby domain and slot representations are integrated in deeper layers.

3.2.4 State Generator

The representations Z^{dst} are used as context-aware representations to decode individual dialogue states. Given a domain index i and slot index j, the feature vector $Z_{dst}[i, j, :] \in \mathbb{R}^d$ is used to generate value of the corresponding (domain, slot) pair. The vector is used as an initial hidden state for an RNN decoder to decode an *inform* slot value. Given the k-th (domain, slot) pair and decoding step l, the output hidden state in each recurrent step h_{kl} is passed through a linear transformation with softmax to obtain output distribution over vocabulary set V.

$$P^{\text{inf}}_{\text{kl}} = \text{Softmax}(h_{kl}W_{\text{inf}}) \in \mathbb{R}^{\|V\|}$$

where $W^{\text{inf}}_{\text{dst}} \in \mathbb{R}^{d_{\text{rnn}}\times\|V\|}$. For *request* slot of k-th (domain,slot) pair, we pass the corresponding vector Z_{dst} vector through a linear layer with sigmoid activation to predict a value of 0 or 1.

$$P^{\text{req}}_{\text{k}} = \text{Sigmoid}(Z^{dst}_k W_{\text{req}})$$

Optimization. The DST is optimized by the cross-entropy loss functions of *inform* and *request* slots:

$$\mathcal{L}_{\text{dst}} = \mathcal{L}_{\text{inf}} + \mathcal{L}_{\text{req}} = \sum_{k=1}^{\|DS\|}\sum_{l=1}^{\|Y_k\|} -\log(P^{\text{inf}}_{\text{kl}}(y_{kl}))$$
$$+ \sum_{k=1}^{\|DS\|} -y_k \log(P^{\text{req}}_k) - (1-y_k)(1-\log(P^{\text{req}}_k))$$

3.3 Joint Dialogue Act and Response Generator

Database Representations. The decoded dialogue states are used to query the database and obtain the number of resulting entities in each domain. Following (Budzianowski et al., 2018), we create a one-hot vector for each domain d: $x^d_{\text{db}} \in \{0, 1\}^6$ and $\sum_i^6 x^d_{\text{db},i} = 1$. Each position of the vector indicates a number or a range of entities. The vectors of all domains are concatenated to create a multi-domain vector $X_{\text{db}} \in \mathbb{R}^{6\times\|D\|}$. We embed this vector as described in Section 3.1 (i.e. with a matrix $E_{\text{db}} \in \mathbb{R}^{2\times d}$), resulting in $Z_{\text{db}} \in \mathbb{R}^{6\times\|D\|\times d}$.

Response→Dialogue Attention. We propose a stacked-attention architecture that sequentially learns dependencies between each token in target responses with each dialogue component representation. First, we obtain $Z^{\text{gen}}_{\text{res}} = \text{Att}(Z_{\text{res}}, Z_{\text{res}}) \in \mathbb{R}^{L_{\text{res}}\times d}$. This attention layer can learn semantics within the target response to construct a more semantically structured sequence. We then use attention to capture dependencies in background information contained in dialogue context and user utterance. The output are $Z^{\text{gen}}_{\text{ctx}} = \text{Att}(Z_{\text{ctx}}, Z^{\text{gen}}_{\text{res}}) \in \mathbb{R}^{L_{\text{res}}\times d}$ and $Z^{\text{gen}}_{\text{utt}} = \text{Att}(Z_{\text{utt}}, Z^{\text{gen}}_{\text{ctx}}) \in \mathbb{R}^{L_{\text{res}}\times d}$ sequentially.

Response→State and DB Attention. To incorporate information of dialogue states and DB results, we apply attention steps to capture dependencies between each response token representation and state or DB representation. Specifically, we first obtain $Z^{\text{gen}}_{\text{dst}} = \text{Att}(Z^{\text{dst}}, Z^{\text{gen}}_{\text{utt}}) \in \mathbb{R}^{L_{\text{res}}\times d}$. In the context-to-text setting, as we directly use the ground-truth dialogue states, we simply replace Z^{dst} with $Z^{\text{curr}}_{\text{st}}$. Then we obtain $Z^{\text{gen}}_{\text{db}} = \text{Att}(Z_{\text{db}}, Z^{\text{gen}}_{\text{dst}}) \in \mathbb{R}^{L_{\text{res}}\times d}$. These attention layers capture the information needed to generate tokens that are towards task completion and supplement the contextual cues obtained in previous attention layers. We let the models to progressively capture these dependencies for N^{gen} times and denote the final output as Z^{gen}. The final output is passed to

a linear layer with softmax activation to decode system responses auto-regressively.

$$P^{\text{res}} = \text{Softmax}(Z^{\text{gen}} W_{\text{gen}}) \in \mathbb{R}^{L_{\text{res}} \times \|V_{\text{res}}\|}$$

Dialogue Act Modeling. We couple response generation with dialogue act modeling by learning a latent variable $Z_{\text{act}} \in \mathbb{R}^d$. We place the vector in the first position of Z_{res}, resulting in $Z_{\text{res+act}} \in \mathbb{R}^{(L_{\text{res}}+1) \times d}$. We then pass this tensor to the same stacked attention layers as above. By adding the latent variable in the first position, we allow our model to semantically condition all downstream tokens from second position, i.e. all tokens in the target response, on this latent variable. The output representation of the latent vector i.e. first row in Z^{gen}, incorporates contextual signals accumulated from all attention layers and is used to predict dialogue acts. We denote this representation as $Z^{\text{gen}}_{\text{act}}$ and pass it through a linear layer to obtain a multi-hot encoded tensor. We apply Sigmoid on this tensor to classify each dialogue act as 0 or 1.

$$P^{\text{act}} = \text{Sigmoid}(Z^{\text{gen}}_{\text{act}} W_{\text{act}}) \in \mathbb{R}^{\|A\|}$$

Optimization. The response generator is jointly trained by the cross-entropy loss functions of generated responses and dialogue acts:

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{res}} + \mathcal{L}_{\text{act}} = \sum_{l=1}^{\|Y_{\text{res}}\|} -\log(P^{\text{res}}_l(y_l))$$
$$+ \sum_{a=1}^{\|A\|} -y_a \log(P^{\text{act}}_a) - (1 - y_a)(1 - \log(P^{\text{act}}_a))$$

4 Experiments

4.1 Dataset

We used the multi-domain dialogue corpus MultiWOZ 2.0 (Budzianowski et al., 2018) as well as MultiWOZ 2.1 (Eric et al., 2019) which includes corrected state labels for the DST task. From the dialogue state annotation of the training data, we identified all possible domains and slots. We identified $\|D\| = 7$ domains and $\|S\| = 30$ slots, including 19 *inform* slots and 11 *request* slots. We also identified $\|A\| = 32$ acts. The corpus includes 8,438 dialogues in the training set and 1,000 in each validation and test set. On average, each dialogue has 1.8 domains and extends over 13 turns. The benchmark also includes an entity DB which can be constructed as a self-contained SQL database engine. We detail additional information of data pre-processing procedures, domains, slots, and entity DBs, in the Appendix A.

4.2 Experiment Setup

We select $d = 256$, $h_{\text{att}} = 8$, $N^{\text{dst}}_S = N^{\text{dst}}_D = N^{\text{gen}} = 3$. We employed dropout (Srivastava et al., 2014) of 0.3 and label smoothing (Szegedy et al., 2016) on target system responses during training. We adopt a teacher-forcing training strategy by simply using the ground-truth inputs of dialogue state of the previous turn and the gold DB representations. During inference in DST and end-to-end tasks, we decode system responses sequentially turn by turn, using the previously decoded state as input in the current turn, and at each turn, using the new predicted state to query DBs. For the context-to-text generation task, ground-truth dialogue states and DBs are used during both training and inference. We train all networks with Adam optimizer (Kingma and Ba, 2015) and a learning rate schedule similarly adopted by (Vaswani et al., 2017). We used batch size 32 and tuned the *warmup_steps* from 10K to 15K training steps. All models are trained up to 30 epochs and the best models are selected based on validation loss. We used a greedy approach to decode all slots and beam search with beam size 5 and a length penalty of 1.0 to decode responses. To evaluate the models, we use the following metrics: (1) DST: Joint Accuracy and Slot Accuracy (Henderson et al., 2014b). (2) Context-to-Text Generation: Inform and Success (Wen et al., 2017) and BLEU score (Papineni et al., 2002).

4.3 Results

DST. We test our state tracker (i.e. using only $\mathcal{L}_{\text{dst}}$) and compare the performance with the baseline models in Table 2 (Refer to the Appendix B for a description of DST baselines). Our model achieves the SOTA performance in MultiWOZ2.1 corpus. Our model can outperform fixed-vocabulary approaches such as HJST and FJST, showing the advantage of generating unique slot values rather than relying on a slot ontology with a fixed set of candidates. DST Reader does not perform as well and we note that many slot values are not easily expressed as a text span in source text inputs. DST approaches that separate domain and slot representations such as TRADE reveal competitive performance. However, our approach has better performance as we adopt a *late fusion* strategy to explicitly obtain more fine-grained contextual de-

Model	Joint Acc.
HJST (Eric et al., 2019)	35.55%
DST Reader (Gao et al., 2019)	36.40%
TSCP (Lei et al., 2018)	37.12%
FJST (Eric et al., 2019)	38.00%
HyST (Goel et al., 2019)	38.10%
TRADE (Wu et al., 2019a)	45.60%
NADST (Le et al., 2020)	49.04%
BDST (Ours)	**49.55%**

Table 2: Evaluation of DST on MultiWOZ2.1

Model	Inform	Success	BLEU
Baseline Budzianowski et al. (2018)	71.29%	60.96%	18.80
TokenMoE (Pei et al., 2019)	75.30%	59.70%	16.81
HDSA (Chen et al., 2019)	82.90%	68.90%	**23.60**
Structured Fusion (Mehri et al., 2019)	82.70%	72.10%	16.34
LaRL (Zhao et al., 2019)	82.78%	**79.20%**	12.80
GPT2 (Budzianowski and Vulić, 2019)	70.96%	61.36%	19.05
DAMD (Zhang et al., 2019)	**89.50%**	75.80%	18.30
DARG (Ours)	87.80%	73.60%	18.80

Table 3: Evaluation of context-to-text task on MultiWOZ2.0.

Model	Joint Acc	Slot Acc	Inform	Success	BLEU
TSCP (L=8) (Lei et al., 2018)	31.64%	95.53%	45.31%	38.12%	11.63
TSCP (L=20) (Lei et al., 2018)	37.53%	96.23%	66.41%	45.32%	15.54
HRED-TS (Peng et al., 2019)	-	-	70.00%	58.00%	17.50
Structured Fusion (Mehri et al., 2019)	-	-	73.80%	58.60%	16.90
DAMD (Zhang et al., 2019)	-	-	**76.30%**	60.40%	16.60
UniConv (Ours)	**50.14%**	**97.30%**	72.60%	**62.90%**	**19.80**

Table 4: Evaluation on MultiWOZ2.1 in the end-to-end setting.

X_{ctx}	B_{t-1}	N_S^{dst}	$N_{\text{D}}^{\text{dst}}$	N^{gen}	$\mathcal{L}_{\text{act}}$	d	h_{att}	Joint Acc.	Slot Acc.	Inform	Success	BLEU
R_{t-1}	✓	3	3	0		256	8	**49.55%**	97.32%	-	-	-
R_{t-1}	✓	3	0	0		256	8	47.91%	97.25%	-	-	-
R_{t-1}	✓	2	2	0		256	8	47.80%	97.22%	-	-	-
R_{t-1}	✓	1	1	0		256	8	46.20%	97.08%	-	-	-
$(U,R)_{1:t-1}$	✓	3	3	0		256	8	49.20%	**97.34%**	-	-	-
R_{t-1}		0	0	3	✓	256	8	-	-	**87.90%**	**72.70%**	18.52
R_{t-1}		0	0	3		256	8	-	-	82.70%	70.60%	18.51
$(U,R)_{1:t-1}$		0	0	3	✓	256	8	-	-	87.14%	71.52%	**18.90**
R_{t-1}		0	0	2	✓	256	8	-	-	81.60%	66.40%	18.48
R_{t-1}		0	0	1	✓	256	8	-	-	77.70%	62.80%	18.50
R_{t-1}	✓	3	3	3	✓	256	8	**50.14%**	**97.30%**	**72.60%**	**62.90%**	19.80
R_{t-1}	✓	3	3	3	✓	128	8	45.70%	97.00%	67.40%	58.30%	**19.90**
R_{t-1}	✓	3	3	3	✓	256	4	47.30%	97.10%	68.70%	57.10%	19.60
R_{t-1}	✓	3	3	3	✓	256	2	45.90%	97.00%	66.10%	55.60%	19.80
R_{t-1}	✓	3	3	3	✓	256	1	43.30%	96.70%	62.30%	52.60%	**19.90**

Table 5: Ablation analysis on the MultiWOZ2.1 in DST (top), context-to-text (middle), and end-to-end (bottom).

pendencies in each domain and slot representation. In this aspect, our model is related to TSCP which decodes output state sequence auto-regressively. However, TSCP attempts to learn domain and slot dependencies implicitly and the model is limited by selecting the maximum output state length (which can vary significantly in multi-domain dialogues).

Context-to-Text Generation. We compare with existing baselines in Table 3 (Refer to the Appendix B for a description of the baseline models). Our model achieves very competitive Inform, Success, and BLEU scores. Compared to TokenMOE, our single model can outperform multiple domain-specific dialogue agents as each attention module can sufficiently learn contextual features of multiple domains. Compared to HDSA which uses a graph structure to represent *acts*, our approach is simpler yet able to outperform HDSA in Inform score. Our work is related to Structured Fusion as we incorporate intermediate representations during decoding. However, our approach does not rely on pre-training individual sub-modules but simultaneously learning both act representations and predicting output tokens. Similarly, our stacked attention architecture can achieve good performance in BLEU score, competitively with a GPT-2 based model, while consistently improve other metrics. For completion, we tested our models on MultiWOZ2.1 and achieved similar results: 87.90% *Inform*, 72.70% *Success*, and 18.52 BLEU score. Future work may further improve *Success* by optimizing the models towards a higher success rate using strategies such as LaRL.

End-to-End. From Table 4, our model outperforms existing baselines in all metrics except the Inform score (See Appendix B for a description of baseline models). In TSCP (Lei et al., 2018), increasing the maximum dialogue state span L from 8 to 20 tokens helps to improve the DST performance, but also increases the training time significantly. Compared with HRED-TS (Peng et al., 2019), our single model generates better responses in all domains without relying on multiple domain-specific teacher models. We also noted that the performance

of DST improves in contrast to the previous DST task. This can be explained as additional supervision from system responses not only contributes to learn a natural response but also positively impact the DST component. We experimented with other baselines along the line of research of sequence-to-sequence framework (Eric and Manning, 2017; Wu et al., 2019b) but could not fully optimize the models due to the large scale of the MultiWOZ benchmark. For example, following GLMP (Wu et al., 2019b), the *restaurant* domain alone has over 1,000 memory tuples of *(Subject, Relation, Object)*.

Ablation. We experiment with several model variants in Table 5 and have the following observations: (1) Using a single-level DST (by considering $S = DS$ and $N_D^{\text{dst}} = 0$) performs worse than the Bi-level DST. Using the dual architecture also improves the latency in each attention layers as typically $\|D\| + \|S\| \ll \|DS\|$. (2) Using B_{t-1} and only the last user utterance as the dialogue context performs as well as using B_{t-1} and a full-length dialogue history. Using only the last user utterance, however, reduces the training time as the number of tokens in a full-length dialogue history is much larger than that of a dialogue state (particularly as the conversation evolves over many turns). (3) We note that removing the loss function to learn the dialogue act latent variable can hurt the generation performance. This reveals the benefit of enforcing a semantic condition on each token of the target response to steer the conversation towards the right direction for task completion. (4) In both state tracker and response generator modules, we note that learning feature representations through deeper attention networks can improve the quality of predicted states and system responses. This is consistent with our DST performance as compared to baseline models of shallow networks. (5) Lastly, our model achieves better performance as the number of attention heads increases, by learning more high-resolution dependencies.

4.4 Domain-dependent Results

DST. For state tracking, the metrics are calculated for domain-specific slots of the corresponding domain at each dialogue turn. We also report the DST separately for multi-domain and single-domain dialogues to evaluate the challenges in multi-domain dialogues and our DST performance gap as compared to single-domain dialogues. From Table 6, our DST performs consistently well in the 3 domains *attraction, restaurant*, and *train* domains. However, the performance drops in the *taxi* and *hotel* domain, significantly in the *taxi* domain. We note that dialogues with the *taxi* domain is usually not single-domain but typically entangled with other domains. Secondly, we observe that there is a significant performance gap of about 10 points absolute score between DST performances in single-domain and multi-domain dialogues. State tracking in multi-domain dialogues is, hence, could be further improved to boost the overall performance.

Domain	Joint Acc	Slot Acc
Multi-domain	48.40%	97.14%
Single-domain	59.63%	98.36%
Attraction	66.76%	98.94%
Hotel	47.86%	97.54%
Restaurant	65.11%	98.68%
Taxi	30.84%	96.86%
Train	63.77%	98.53%

Table 6: DST results on MultiWOZ2.1 by domains.

Context-to-Text Generation For this task, we calculated the metrics for single-domain dialogues of the corresponding domain (as Inform and Success are computed per dialogue rather than per turn). We do not report the *Inform* metric of the *taxi* domain because no DB was available for this domain. From Table 7, we observe some performance gap between *Inform* and *Success* scores on multi-domain dialogues and single-domain dialogues. However, in terms of BLEU score, our model performs better with multi-domain dialogues. This could be caused by the data bias in MultiWOZ corpus as the majority of dialogues in this corpus is multi-domain. Hence, our models capture the semantics of multi-domain dialogue responses better than single-domain responses. For domain-specific results, we note that our models perform not as well as other domains in *attraction* and *taxi* domains in terms of *Success* score.

Domain	Inform	Success	BLEU
Multi-domain	85.01%	68.86%	18.68
Single-domain	97.79%	85.84%	17.62
Attraction	91.67%	66.67%	19.17
Hotel	97.01%	91.04%	16.55
Restaurant	96.77%	88.71%	19.88
Taxi	-	78.85%	13.85
Train	99.10%	87.88%	18.14

Table 7: Context-to-text generation results on MultiWOZ2.1. by domains.

We conduct qualitative analysis and the insights can be seen in Appendix C.

5 Conclusion

We proposed UniConv, a novel unified neural architecture of conversational agents for Multi-domain Task-oriented Dialogues, which jointly trains (1) a Bi-level State Tracker to capture dependencies in both domain and slot levels simultaneously, and (2) a Joint Dialogue Act and Response Generator to model dialogue act latent variable and semantically conditions output responses with contextual cues. The promising performance of UniConv on the MultiWOZ benchmark (including three tasks: DST, context-to-text generation, and end-to-end dialogues) validates the efficacy of our method.

References

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

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22, Hong Kong. Association for Computational Linguistics.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, 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 (EMNLP)*.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709, Florence, Italy. Association for Computational Linguistics.

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*.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49, Saarbrücken, Germany. Association for Computational Linguistics.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473, Valencia, Spain. Association for Computational Linguistics.

Revanth Gangi Reddy, Danish Contractor, Dinesh Raghu, and Sachindra Joshi. 2019. Multi-level memory for task oriented dialogs. 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 3744–3754, Minneapolis, Minnesota. Association for Computational Linguistics.

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, Stockholm, Sweden. Association for Computational Linguistics.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. HyST: A Hybrid Approach for Flexible and Accurate Dialogue State Tracking. In *Proc. Interspeech 2019*, pages 1458–1462.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

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, Florence, Italy. Association for Computational Linguistics.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 1437–1447.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 733–743, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Interspeech 2017*.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478. Association for Computational Linguistics.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jiahuan Pei, Pengjie Ren, and Maarten de Rijke. 2019. A modular task-oriented dialogue system using a neural mixture-of-experts. *arXiv preprint arXiv:1907.05346*.

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

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

Shuke Peng, Xinjing Huang, Zehao Lin, Feng Ji, Haiqing Chen, and Yin Zhang. 2019. Teacher-student framework enhanced multi-domain dialogue generation. *arXiv preprint arXiv:1908.07137*.

Libo Qin, Yijia Liu, Wanxiang Che, Haoyang Wen, Yangming Li, and Ting Liu. 2019. Entity-consistent end-to-end task-oriented dialogue system with KB retriever. 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 133–142, Hong Kong, China. Association for Computational Linguistics.

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

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*, volume 2, 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, Hong Kong, China. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lei Shu, Piero Molino, Mahdi Namazifar, Hu Xu, Bing Liu, Huaixiu Zheng, and Gokhan Tur. 2019. Flexibly-structured model for task-oriented dialogues. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 178–187, Stockholm, Sweden. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Shang-Yu Su, Kai-Ling Lo, Yi Ting Yeh, and Yun-Nung Chen. 2018. Natural language generation by hierarchical decoding with linguistic patterns. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 61–66, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. Conditional generation and snapshot learning in neural dialogue systems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2153–2162, Austin, Texas. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

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

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677, Vancouver, Canada. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019a. 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, Florence, Italy. Association for Computational Linguistics.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019b. Global-to-local memory pointer networks for task-oriented dialogue. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Semih Yavuz, Abhinav Rastogi, Guan-Lin Chao, and Dilek Hakkani-Tur. 2019. DeepCopy: Grounded response generation with hierarchical pointer networks. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 122–132, Stockholm, Sweden. Association for Computational Linguistics.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2019. Task-oriented dialog systems that consider multiple appropriate responses under the same context. *arXiv preprint arXiv:1911.10484*.

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 27–36, Saarbrücken, Germany. Association for Computational Linguistics.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. 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 1208–1218, Minneapolis, Minnesota. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *ACL*.

A Data Pre-processing

First, we delexicalize each target system response sequence by replacing the matched entity attribute that appears in the sequence to the canonical tag $\langle domain\_slot \rangle$. For example, the original target response 'the train id is tr8259 departing from cambridge' is delexicalized into 'the train id is *train_id* departing from *train_departure*'. We use the provided entity databases (DBs) to match potential attributes in all target system responses. To construct dialogue history, we keep the original version of all text, including system responses of previous turns, rather than the delexicalized form. We split all sequences of dialogue history, user utterances of the current turn, dialogue states, and delexicalized target responses, into case-insensitive tokens. We share the embedding weights of all source sequences, including dialogue history, user utterance, and dialogue states, but use a separate embedding matrix to encode the target system responses.

We summarize the number of dialogues in each domain in Table 8. For each domain, a dialogue is selected as long as the whole dialogue (i.e. single-domain dialogue) or parts of the dialogue (i.e. in multi-domain dialogue) is involved with the domain. For each domain, we also build a set of possible *inform* and *request* slots using the dialogue state annotation in the training data. The details of slots and database in each domain can be seen in Table 9. The DBs of 3 domains *taxi, police*, and *hospital* are not available as part of the benchmark.

Domain	#dialogues		
	train	val	test
Restaurant	3,817	438	437
Hotel	3,387	416	394
Attraction	2,718	401	396
Train	3,117	484	495
Taxi	1,655	207	195
Police	245	0	0
Hospital	287	0	0

Table 8: Summary of MultiWOZ dataset (Budzianowski et al., 2018) by domain

B Baselines

We describe our baseline models in DST, context-to-text generation, and end-to-end dialogue tasks.

B.1 DST

FJST and **HJST** (Eric et al., 2019). These models adopt a fixed-vocabulary DST approach. Both models include encoder modules (either bidirectional LSTM or hierarchical LSTM) to encode the dialogue history. The models pass the context hidden states to separate linear transformation to obtain final vectors to predict individual slots separately. The output vector is used to measure a score of each candidate from a predefined candidate set.

DST Reader (Gao et al., 2019). This model considers the DST task as a reading comprehension task and predicts each slot as a span over tokens within dialogue history. DST Reader utilizes attention-based neural networks with additional modules to predict slot type and carryover probability.

TSCP (Lei et al., 2018). The model adopts a sequence-to-sequence framework with a pointer network to generate dialogue states. The source sequence is a combination of the last user utterance, dialogue state of the previous turn, and user utterance. To compare with TSCP in a multi-domain task-oriented dialogue setting, we adapt the model to multi-domain dialogues by formulating the dialogue state of the previous turn similarly as our models. We reported the performance when the maximum length of the output dialogue state sequence L is set to 20 tokens (original default parameter is 8 tokens but we expect longer dialogue state in MultiWOZ benchmark and selected 20 tokens).

HyST (Goel et al., 2019). This model combines the advantage of fixed-vocabulary and open-vocabulary approaches. The model uses an open-vocabulary approach in which the set of candidates of each slot is constructed based on all word n-grams in the dialogue history. Both approaches are applied in all slots and depending on their performance in the validation set, the better approach is used to predict individual slots during test time.

TRADE (Wu et al., 2019a). The model adopts a sequence-to-sequence framework with a pointer network to generate individual slot token-by-token. The prediction is additionally supported by a slot gating component that decides whether the slot is "none", "dontcare", or "generate". When the gate of a slot is predicted as "generate", the model will generate value as a natural output sequence for that slot.

NADST (Le et al., 2020). This is the current state-of-the-art model on the MultiWOZ2.1 dataset. The model proposes a non-autoregressive approach for dialogue state tracking which enables learning dependencies between domain-level and slot-level representations as well as token-level representations of slot values.

Domain	Slots	#entities	DB attributes
Restaurant	inf_area, inf_food, inf_name, inf_pricerange, inf_bookday, inf_bookpeople, inf_booktime, req_address, req_area, req_food, req_phone, req_postcode	110	id, address, area, food, introduction, name, phone, postcode, pricerange, signature, type
Hotel	inf_area, inf_internet, inf_name, inf_parking, inf_pricerange, inf_stars, inf_type, inf_bookday, inf_bookpeople, inf_bookstay, req_address, req_area, req_internet, req_parking, req_phone, req_postcode, req_stars, req_type	33	id, address, area, internet, parking, single, double, family, name, phone, postcode, pricerange', takesbookings, stars, type
Attraction	inf_area, inf_name, inf_type, req_address, req_area, req_phone, req_postcode, req_type	79	id, address, area, entrance, name, phone, postcode, pricerange, openhours, type
Train	inf_arriveBy, inform_day, inf_departure, inf_destination, inf_leaveAt, inf_bookpeople, req_duration, req_price	2,828	trainID, arriveBy, day, departure, destination, duration, leaveAt, price
Taxi	inf_arriveBy, inf_departure, inf_destination, inf_leaveAt, req_phone	-	-
Police	inf_department, req_address, req_phone, req_postcode	-	-
Hospital	req_address, req_phone, req_postcode	-	-

Table 9: Summary of slots and DB details by domain in the MultiWOZ dataset (Budzianowski et al., 2018)

B.2 Context-to-Text Generation

Baseline. (Budzianowski et al., 2018) provides a baseline for this setting by following the sequence-to-sequence model (Sutskever et al., 2014). The source sequence is all past dialogue turns and the target sequence is the system response. The initial hidden state of the RNN decoder is incorporated with additional signals from the dialogue states and database representations.

TokenMoE (Pei et al., 2019). TokenMoE refers to Token-level Mixture-of-Expert model. The model follows a modularized approach by separating different components known as expert bots for different dialogue scenarios. A dialogue scenario can be dependent on a domain, a type of dialogue act, etc. A chair bot is responsible for controlling expert bots to dynamically generate dialogue responses.

HDSA (Chen et al., 2019). This is the current state-of-the-art in terms of Inform and BLEU score in the context-to-text generation setting in MultiWOZ2.0. HDSA leverages the structure of dialogue acts to build a multi-layer hierarchical graph. The graph is incorporated as an inductive bias in a self-attention network to improve the semantic quality of generated dialogue responses.

Structured Fusion (Mehri et al., 2019). This approach follows a traditional modularized dialogue system architecture, including separate components for NLU, DM, and NLG. These components are pre-trained and combined into an end-to-end system. Each component output is used as a structured input to other components.

LaRL (Zhao et al., 2019). This model uses a latent dialogue action framework instead of handcrafted dialogue acts. The latent variables are learned using unsupervised learning with stochastic variational inference. The model is trained in a reinforcement learning framework whereby the parameters are trained to yield a better Success rate. The model is the current state-of-the-art in terms of Success metric.

GPT2 (Budzianowski and Vulić, 2019). Unsupervised pre-training language models have significantly improved machine learning performance in many NLP tasks. This baseline model leverages the power of a pre-trained model (Radford et al., 2019) and adapts to the context-to-text generation setting in task-oriented dialogues. All input components, including dialogue state and database state, are transformed into raw text format and concatenated as a single sequence. The sequence is used as input to a pre-trained GPT-2 model which is then fine-tuned with MultiWOZ data.

DAMD (Zhang et al., 2019). This is the current state-of-the-art model for context-to-text generation task in MultiWOZ 2.1. This approach augments training data with multiple responses of similar context. Each dialogue state is mapped to multiple valid dialogue acts to create additional state-act pairs.

B.3 End-to-End

TSCP (Lei et al., 2018). In addition to the DST task, we evaluate TSCP as an end-to-end dialogue system that can do both DST and NLG. We adapt the models to the multi-domain DST setting as

described in Section B.1 and keep the original response decoder. Similar to the DST component, the response generator of TSCP also adopts a pointer network to generate tokens of the target system responses by copying tokens from source sequences. In this setting, we test TSCP with two settings of the maximum length of the output dialogue state sequence: $L = 8$ and $L = 20$.

HRED-TS (Peng et al., 2019). This model adopts a teacher-student framework to address multi-domain task-oriented dialogues. Multiple teacher networks are trained for different domains and intermediate representations of dialogue acts and output responses are used to guide a universal student network. The student network uses these representations to directly generate responses from dialogue context without predicting dialogue states.

C Qualitative Analysis

We examine an example of dialogue in the test data and compare our predicted outputs with the baseline TSCP ($L = 20$) (Lei et al., 2018) and the ground truth. From Figure 4, we observe that both our predicted dialogue state and system response are more correct than the baseline. Specifically, our dialogue state can detect the correct *type* slot in the *attraction* domain. As our dialogue state is correctly predicted, the queried results from DB is also more correct, resulting in better response with the right information (i.e. 'no attraction available'). In Figure 5, we show the visualization of domain-level and slot-level attention on the user utterance. We notice important tokens of the text sequences, i.e. 'entertainment' and 'close to', are attended with higher attention scores. Besides, at domain-level attention, we find a potential additional signal from the token 'restaurant', which is also the domain from the previous dialogue turn. We also observe that attention is more refined throughout the neural network layers. For example, in the domain-level processing, compared to the 2^{nd} layer, the 4^{th} layer attention is more clustered around specific tokens of the user utterance.

In Table 10 and 11, we reported the complete output of this example dialogue. Overall, our dialogue agent can carry a proper dialogue with the user throughout the dialogue steps. Specifically, we observed that our model can detect new domains at dialogue steps where the domains are introduced e.g. *attraction* domain at the 5^{th} turn and *taxi* domain at the 8^{th} turn. The dialogue agent can also detect some of the co-references among the domains. For example, at the 5^{th} turn, the dialogue agent can infer the slot *area* for the new domain *attraction* as the user mentioned 'close the restaurant'. We noticed that that at later dialogue steps such as the 6^{th} turn, our decoded dialogue state is not correct possibly due to the incorrect decoded dialogue state in the previous turn, i.e. 5^{th} turn.

In Figure 2 and 3, we plotted the Joint Goal Accuracy and BLEU metrics of our model by dialogue turn. As we expected, the Joint Accuracy metric tends to decrease as the dialogue history extends over time. The dialogue agent achieves the highest accuracy in state tracking at the 1^{st} turn and gradually reduces to zero accuracy at later dialogue steps, i.e. 15^{th} to 18^{th} turns. For response generation performance, the trend of BLEU score is less obvious. The dialogue agent obtains the highest BLEU scores at the 3^{rd} turn and fluctuates between the 2^{nd} and 13^{th} turn.

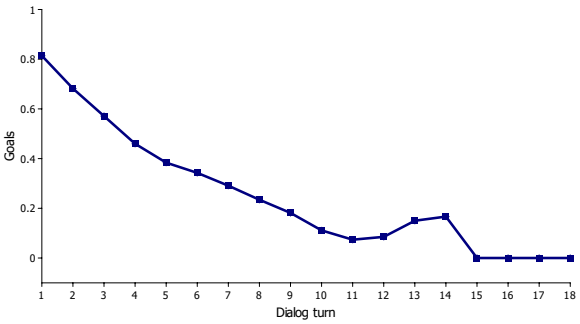

Figure 2: Joint Accuracy metric by dialogue turn in the test data.

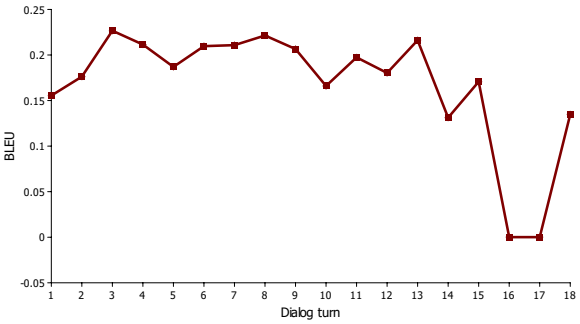

Figure 3: BLEU4 metric by dialogue turn in the test data.

R_4: all set . your reference number is k2bo09vq .
U_5: thanks . i am also looking for some entertainment close to the restaurant . any suggestions ?

B_5: *{restaurant: {area: center, name: dojo noodle bar, pricerange: cheap}, attraction: { area: center, type: entertainment}}*
R_5: i am sorry i **do not have** any attractions meeting the criteria you listed . is there another type of attraction or area you would like me to search ?

B_5^{tscp}: *{restaurant: {area: center, pricerange: cheap}, attraction: { area: center}}*
R_5^{tscp}: there are **many** attractions in the restaurant_area . do you have a preference?

B_5^{ours}: *{restaurant: {area: center, pricerange: cheap}, attraction: { area: center, type: entertainment}}*
R_5^{ours}: there are **no** attraction_type attractions in the restaurant_area of town . would you like to try another area ?

Figure 4: Example dialogue with the input system response R_{t-1} and current user utterance U_t, and the output state BS_t and system response R_t. Compared with TSCP, our dialogue state and response are more correct and closer to the ground truth.

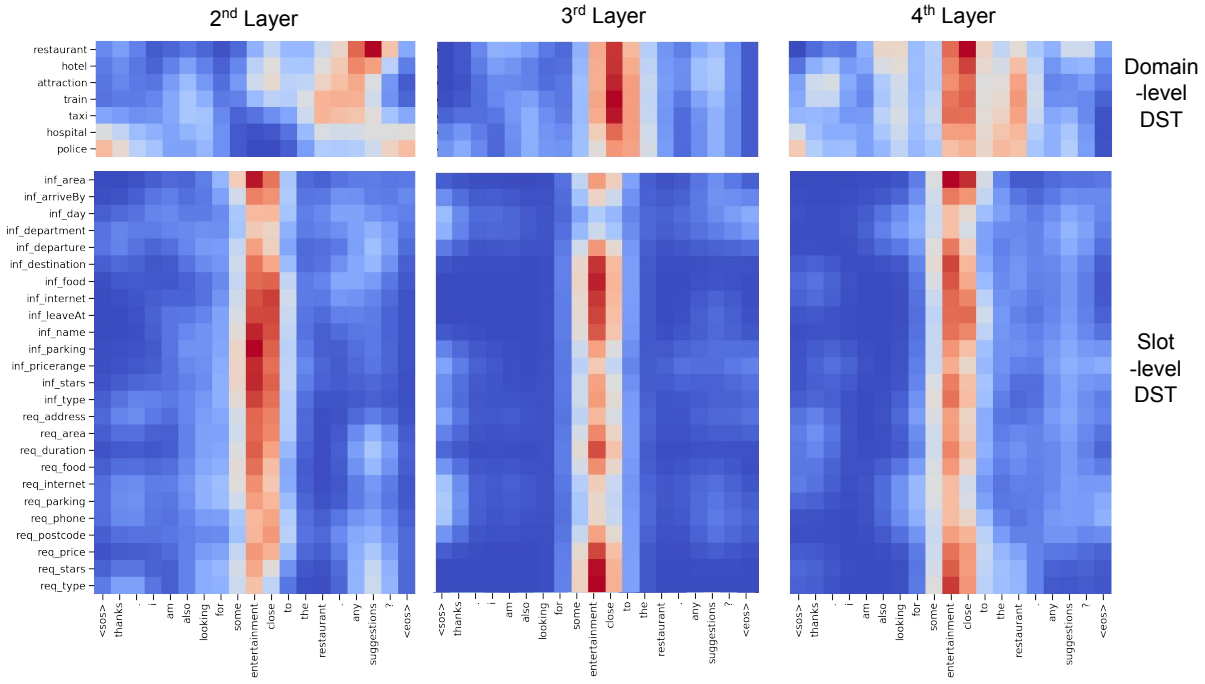

Figure 5: Visualization of attention to the user utterance sequence at slot-level (lower right) and domain-level (upper right) is also included. More red denotes higher attention score between domain or slot representation and token representation. Best viewed in color.

Turn	Input Dialogue	Ground-truth State and Response	Our Predicted State and Response
	R_0: <blank>	B_1: {restaurant: {area: centre, pricerange: cheap}}	B_1: {restaurant: {area: centre, pricerange: cheap}}
1	U_1: i am looking for a cheap restaurant in the center of the city	R_1: do you have any specific type of food you would like ?	R_1: there are 15 restaurant_pricerange restaurant_type s in the restaurant_area of town . do you have a preference ?
	U_1: do you have any specific type of food you would like ?	B_2: {restaurant: {area: centre, pricerange: cheap}}	B_2: {restaurant: {area: centre, pricerange: cheap}}
2	U_2: no , i am not picky as long as the price -s are low .	R_2: there is a restaurant_pricerange restaurant_food restaurant_type called the restaurant_name located in the restaurant_area of town . would you like to book a table ?	R_2: how about restaurant_name ? it s an restaurant_pricerange restaurant_food restaurant_type located in the restaurant_area of town .
	R_2: there is a cheap chinese restaurant called the dojo noodle bar located in the centre of town . would you like to book a table ?	B_3: {restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}}	B_2: {restaurant: {area: centre, pricerange: cheap}}
3	U_3: yes please , for 8 people at 1830 on thursday .	R_3: i am sorry but restaurant_name is solidly booked at that time . i can try a different time or day for you .	R_3: booking was successful . the table will be reserved for 15 minutes . reference number is restaurant_reference .
	R_3: i am sorry but dojo noodle bar is solidly booked at that time . i can try a different time or day for you	B_4: {restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}}	B_4: {restaurant: {area: centre, pricerange: cheap}}
4	U_4: can you try to book it at 1730 .	R_4: all set . your reference number is restaurant_reference .	R_4: booking was successful . the table will be reserved for 15 minutes . reference number is restaurant_reference .
	R_4: all set . your reference number is k2bo09vq .	B_5: {restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: entertainment}}	B_5: {restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: entertainment}}
5	U_5: thanks . i am also looking for some entertainment close to the restaurant . any suggestions ?	R_5: i am sorry i do not have any attractions meeting the criteria you listed . is there another type of attraction or area you would like me to search ?	R_5: there are no attraction_type attractions in the restaurant_area of town . would you like to try another area ?
	R_5: i am sorry i do not have any attractions meeting the criteria you listed . is there another type of attraction or area you would like me to search ?	B_6:{restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: architecture}}	B_6:{restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: architecture}}
6	U_6: why do not you try an architectural attraction , instead .	R_6: attraction_name looks good , would you like to head there ?	R_6: attraction_name is a great attraction_type in the center of town . would you like more information ?
	R_6: all saints church looks good , would you like to head there ?	B_7:{restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: architecture}}	B_7:{restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: architecture}}
7	U_7: yes please . can you let me know what the entrance fee is too ?	R_7: its entrance fee is attraction_pricerange .	R_7: the entrance fee is attraction_pricerange . is there anything else i can help you with ?
	R_7: its entrance fee is free .	B_8:{restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: architecture}, taxi: {departure: dojo noodle bar, destination: all saints church}}	B_8:{restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: architecture}, taxi: {destination: dojo noodle bar}}
8	U_8: i also need to book a taxi between the restaurant and the church .	R_8: what time would you like the taxi to pick you up from restaurant_name ?	R_8: what time would you like to leave or arrive by ?

Table 10: Complete turns of an example multi-domain dialogue with each turn including the input of past system response R_{t-1} and current user utterance U_t, and the predicted dialogue state B_t and system response U_t. The dialogue consists of 11 turns in total and extends across 3 domains sequentially: *restaurant*, *attraction*, and *taxi*. Dialogue continues in Table 11.

Turn	Input Dialogue	Ground-truth State and Response	Our Predicted State and Response
	R_8: what time would you like the taxi to pick you up from dojo noodle bar ?	B_9: {restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: architecture}, taxi: {departure: dojo noodle bar, destination: all saints church, leaveAt: 1830}}	B_9:{restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: architecture}, taxi: {destination: dojo noodle bar}}
9	U_9: 1830 , please .	R_9: your taxi is booked . the car is a taxi_type and the contact number is taxi_phone . is there anything else i can help you with ?	R_9: booking completed ! booked car type taxi_type contact number taxi_phone can i help you with anything else ?
	R_9: your taxi is booked . the car is a blue bmw and the contact number is 07239644669 . is there anything else i can help you with ?	B_{10}: {restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: architecture}, taxi: {departure: dojo noodle bar, destination: all saints church, leaveAt: 1830}}	B_{10}: {restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: architecture}, taxi: {destination: dojo noodle bar}}
10	U_{10}: that s it . thanks .	R_{10}: awesome , enjoy your trip !	R_{10}: you are welcome . have a great day !
	R_{10}: awesome , enjoy your trip !	B_{11}: {restaurant: {area: centre, pricerange: cheap, name: dojo noodle bar}, attraction: {area: centre, type: architecture}, taxi: {departure: dojo noodle bar, destination: all saints church, leaveAt: 1830}}	B_{11}: {restaurant: {area: centre, pricerange: cheap}, attraction: {area: centre, type: architecture}, taxi: {destination: dojo noodle bar}}
11	U_{11}: thanks ! i will definitely try to .	R_{11}: great ! goodbye and have a nice day .	R_{11}: thank you for using our services . have a great day .

Table 11: Dialogue continues from Table 10. Each turn includes the input of past system response S_{t-1} and current user utterance U_t, and the predicted dialogue dialogue state BS_t and system response S_t. The dialogue consists of 11 turns in total and extends across 3 domains sequentially: restaurant, attraction, and taxi.