NON-AUTOREGRESSIVE DIALOG STATE TRACKING

Hung Le ‡∗, Richard Socher †, Steven C.H. Hoi †
† Salesforce Research
{rsocher,shoi}@salesforce.com
‡ Singapore Management University
hungle.2018@phdcs.smu.edu.sg

ABSTRACT

Recent efforts in Dialogue State Tracking (DST) for task-oriented dialogues have progressed toward open-vocabulary or generation-based approaches where the models can generate slot value candidates from the dialogue history itself. These approaches have shown good performance gain, especially in complicated dialogue domains with dynamic slot values. However, they fall short in two aspects: (1) they do not allow models to explicitly learn signals across domains and slots to detect potential dependencies among (domain, slot) pairs; and (2) existing models follow auto-regressive approaches which incur high time cost when the dialogue evolves over multiple domains and multiple turns. In this paper, we propose a novel framework of Non-Autoregressive Dialog State Tracking (NADST) which can factor in potential dependencies among domains and slots to optimize the models towards better prediction of dialogue states as a complete set rather than separate slots. In particular, the non-autoregressive nature of our method not only enables decoding in parallel to significantly reduce the latency of DST for real-time dialogue response generation, but also detect dependencies among slots at token level in addition to slot and domain level. Our empirical results show that our model achieves the state-of-the-art joint accuracy across all domains on the MultiWOZ 2.1 corpus, and the latency of our model is an order of magnitude lower than the previous state of the art as the dialogue history extends over time.

1 INTRODUCTION

In task-oriented dialogues, a dialogue agent is required to assist humans for one or many tasks such as finding a restaurant and booking a hotel. As a sample dialogue shown in Table 1, each user utterance typically contains important information identified as slots related to a dialogue domain such as attraction-area and train-day. A crucial part of a task-oriented dialogue system is Dialogue State Tracking (DST), which aims to identify user goals expressed during a conversation in the form of dialogue states. A dialogue state consists of a set of (slot, value) pairs e.g. (attraction-area, centre) and (train-day, tuesday). Existing DST models can be categorized into two types: fixed- and open-vocabulary. Fixed vocabulary models assume known slot ontology and generate a score for each candidate of (slot, value) (Ramadan et al., 2018; Lee et al., 2019). Recent approaches propose open-vocabulary models that can generate the candidates, especially for slots such as entity names and time, from the dialogue history (Lei et al., 2018, Wu et al., 2019).

Most open-vocabulary DST models rely on autoregressive encoders and decoders, which encode dialogue history sequentially and generate token $t_i$ of individual slot value one by one conditioned on all previously generated tokens $t_{[1:i-1]}$. For downstream tasks of DST that emphasize on low latency (e.g. generating real-time dialogue responses), auto-regressive approaches incur expensive time cost as the ongoing dialogues become more complex. The time cost is caused by two major components: length of dialogue history i.e. number of turns, and length of slot values. For complex dialogues extended over many turns and multiple domains, the time cost will increase significantly in both encoding and decoding phases.

Similar problems can be seen in the field of Neural Machine Translation (NMT) research where a long piece of text is translated from one language to another. Recent work has tried to improve the

*All work was done while the first author was a research intern at Salesforce Research Asia.
Our work is related to two research areas: dialogue state tracking and non-autoregressive decoding.

2.1 Dialogue State Tracking

Dialogue State Tracking (DST) is an important component in task-oriented dialogues, especially for dialogues with complex domains that require fine-grained tracking of relevant slots. Traditionally,
DST is coupled with Natural Language Understanding (NLU). NLU output as tagged user utterances is input to DST models to update the dialogue states turn by turn (Kurata et al., 2016; Shi et al., 2016; Rastogi et al., 2017). Recent approaches combine NLU and DST to reduce the credit assignment problem and remove the need for NLU (Mrkšić et al., 2017; Xu & Hu, 2018; Zhong et al., 2018). Within this body of research, Goel et al. (2019) differentiates two DST approaches: fixed- and open-vocabulary. Fixed-vocabulary approaches are usually retrieval-based methods in which all candidate pairs of (slot, value) from a given slot ontology are considered and the models predict a probability score for each pair (Henderson et al., 2014c; Ramadan et al., 2018; Lee et al., 2019). Recent work has moved towards open-vocabulary approaches that can generate the candidates based on input text i.e. dialogue history (Lei et al., 2018; Gao et al., 2019; Wu et al., 2019). Our work is more related to these models, but different from most of the current work, we explicitly consider dependencies among slots and domains to decode dialogue state as a complete set.

### 2.2 Non-autoregressive Decoding

Most of prior work in non- or semi-autoregressive decoding methods are used for NMT to address the need for fast translation. Schwenk (2012) proposes to estimate the translation model probabilities of a phase-based NMT system. Libovický & Helcl (2018) formulates the decoding process as a sequence labeling task by projecting source sequence into a longer sequence and applying CTC loss (Graves et al., 2006) to decode the target sequence. Wang et al. (2019) adds regularization terms to NAT models (Gu et al., 2018) to reduce translation errors such as repeated tokens and incomplete sentences. Ghazvininejad et al. (2019) uses a non-autoregressive decoder with masked attention to decode target sequences over multiple generation rounds. A common challenge in non-autoregressive NMT is the large number of sequential latent variables, e.g., fertility sequences (Gu et al., 2018) and projected target sequences (Libovický & Helcl, 2018). These latent variables are used as supporting signals for non- or semi-autoregressive decoding. We reformulate dialogue state as a structured sequence with sub-sequences defined as a concatenation of slot values. This form of dialogue state can be inferred easily from the dialogue state annotation itself whereas such supervision information is not directly available in NMT. The lower semantic complexity of slot values as compared to long sentences in NMT makes it easier to adopt non-autoregressive approaches into DST. According to our review, we are the first to apply a non-autoregressive framework for generation-based DST. Our approach allows joint state tracking across slots, which results in better performance and an order of magnitude lower latency during inference.

### 3 Approach

Our NADST model is composed of three parts: encoders, fertility decoder, and state decoder, as shown in Figure 1. The input includes the dialogue history \( X = (x_1, …, x_N) \) and a sequence of applicable (domain, slot) pairs \( X_{ds} = ((d_1, s_1), …, (d_G, s_H)) \), where \( G \) and \( H \) are the total numbers of domains and slots, respectively. The output is the corresponding dialogue states up to the current dialogue history. Conventionally, the output of dialogue state is denoted as tuple (slot, value) or (domain-slot, value) for multi-domain dialogues. We reformulate the output as a concatenation of slot values \( Y^{d_i, s_j}: Y = (Y^{d_1, s_1}, …, Y^{d_J, s_J}) = (y_1, y_2, …, y_1, y_2, …, y_1, y_2, …, y_1, y_2, …) \) where \( I \) and \( J \) are the numbers of domains and slots in the output dialogue state, respectively.

First, the encoders use token-level embedding and positional encoding to encode the input dialogue history and (domain, slot) pairs into continuous representations. The encoded domains and slots are then input to stacked self-attention and feed-forward network to obtain relevant signals across dialogue history and (domain, slot) pairs into continuous representations. The output of fertility decoder is defined as a sequence: \( Y_{fert} = Y_f^{d_1, s_1}, …, Y_f^{d_G, s_H} \) where \( Y_f^{d_i, s_h} \in \{0, \text{max( SlotLength) } \} \). For example, for the MultiWOZ dataset in our experiments, we have \( \text{max( SlotLength) } = 9 \) according to the training data. We follow Wu et al. (2019) and projected target sequences (Libovický & Helcl, 2018) to add a slot gating mechanism as an auxiliary prediction. Each gate is restricted to 3 possible values: “none”, “dontcare” and “generate”. They are used to form higher-level classification signals to support fertility decoding process. The gate output is defined as a sequence: \( Y_{gate} = Y_g^{d_1, s_1}, …, Y_g^{d_G, s_H} \).

The predicted fertilities are used to form an input sequence to the state decoder for non-autoregressive decoding. The sequence includes sub-sequences of \( (d_g, s_h) \) repeated by \( Y_f^{d_g, s_h} \) times...
and concatenated sequentially: $X_{ds \times fert} = ((d_1, s_1), \ldots, (d_H, s_H))$, and $\|X_{ds \times fert}\| = \|Y\|$. The decoder projects this sequence through attention layers with dialogue history. During this decoding process, we maintain a memory of hidden states of dialogue history. The output from the state decoder is used as a query to attend on this memory and copy tokens from the dialogue history to generate a dialogue state.

Following [Let et al. (2018)], we incorporate information from previous dialogue turns to predict current turn state by using a partially delexicalized dialogue history $X_{del} = (x_{1,del}, \ldots, x_{N,del})$ as an input of the model. The dialogue history is delexicalized till the last system utterance by removing real-value tokens that match the previously decoded slot values to tokens expressed as domain-slot. Given a token $x_n$ and the current dialogue turn $t$, the token is delexicalized as follows:

$$x_{n,del} = delex(x_n) = \begin{cases} x_n, & \text{if } x_n \subset \hat{Y}_{t-1}. \\ \text{domain}_{idx} \text{+ slot}_{idx}, & \text{otherwise.} \end{cases}$$

(1)

$$\text{domain}_{idx} = X_{ds \times fert}[\text{idx}][0], \quad \text{slot}_{idx} = X_{ds \times fert}[\text{idx}][1], \quad \text{idx} = \text{Index}(x_n, \hat{Y}_{t-1})$$

(2)

For example, the user utterance “I look for a cheap hotel” is delexicalized to “I look for a hotel_price range” if the slot hotel_price range is predicted as “cheap” in the previous turn. This approach makes use of the delexicalized form of dialogue history while not relying on an NLU module as we utilize the predicted state from DST model itself. In addition to the belief state, we also use the system action in the previous turn to delexicalize the dialog history in a similar manner, following prior work [Rastogi et al. 2017; Zhong et al. 2018; Goel et al. 2019].

Figure 1: Our NADST has 3 key components: encoders (“red”), fertility decoder (“blue”), and state decoder (“green”). (i) Encoders encode sequences of dialogue history, delexicalized dialogue history, and domain and slot tokens into continuous representations; (ii) Fertility Decoder has 3 attention mechanisms to learn potential dependencies across (domain, slot) pairs in combination with dialogue history. The output is used to generate fertilities and slot gates; and (iii) State Decoder receives the input sequence including sub-sequences of (domain, slot)×fertility to decode a complete dialogue state sequence as concatenation of component slot values. For simplicity, we do not show feedforward, residual connection, and layer-normalization layers in the figure. Best viewed in color.

Published as a conference paper at ICLR 2020
3.1 Encoders

An encoder is used to embed dialogue history $X$ into a sequence of continuous representations $Z = (z_1, ..., z_N) \in \mathbb{R}^{N \times d}$. Similarly, partially delexicalized dialogue history $X_{del}$ is encoded to continuous representations $Z_{del} \in \mathbb{R}^{N \times d}$. We store the encoded dialogue history $Z$ in memory which will be passed to a pointer network to copy words for dialogue state generation. This helps to address the OOV challenge as shown in See et al. [2017] Wu et al. [2019]. We also encode each (domain, slot) pair into continuous representation $z_{ds} \in \mathbb{R}^d$ as input to the decoders. Each vector $z_{ds}$ is used to store contextual signals for slot and fertility prediction during the decoding process.

**Context Encoder.** Context encoder includes a token-level trainable embedding layer and layer normalization [Ba et al., 2016]. The encoder also includes a positional encoding layer which follows sine and cosine functions [Vaswani et al., 2017]. An element-wise summation is used to combine the token-level vectors with positional encoded vectors. We share the embedding weights to embed input and delexicalized dialogue history. The embedding weights are also shared to encode input to both fertility decoder and state decoder. The final embedding of $X$ and $X_{del}$ is defined as:

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

$$Z_{del} = Z_{emb, del} + PE(X_{del}) \in \mathbb{R}^{N \times d}$$

**Domain and Slot Encoder.** Each (domain, slot) pair is encoded by using two separate embedding vectors of the corresponding domain and slot. Each domain $g$ and slot $h$ is embedded into a continuous representation $z_g$ and $z_h \in \mathbb{R}^d$. The final vector is combined by element-wise summation:

$$z_{ds} = z_g + z_h \in \mathbb{R}^d$$

We share the embedding weights to embed domain and slot tokens in both fertility decoder and state decoder. However, for input to state decoder, we inject sequential information into the input $X_{ds \times fert}$ to factor in position-wise information to decode target state sequence. In summary, $X_{ds}$ and $X_{ds \times fert}$ is encoded as following:

$$Z_{ds} = Z_{emb, ds} = z_{d_1, s_1} \oplus ... \oplus z_{d_g, s_h}$$

$$Z_{ds \times fert} = Z_{emb, ds \times fert} + PE(X_{ds \times fert})$$

$$Z_{emb, ds \times fert} = (z_{d_1, s_1})^Y_{dG, sH} \oplus ... \oplus (z_{d_g, s_h})^Y_{dG, sH}$$

where $\oplus$ denotes concatenation operation. Note that different from a typical decoder input in Transformer, we do not shift the input sequences to both fertility decoder and state decoder by one position as we consider non-autoregressive decoding process in both modules. Therefore, all output tokens are generated in position $i$ based on all remaining positions of the sequence i.e. $1, ..., i - 1, i + 1, ... ||X_{ds}||$ in fertility decoder and $1, ..., i - 1, i + 1, ... ||X_{ds \times fert}||$ in state decoder.

3.2 Fertility Decoder

Given the encoded dialogue history $Z$, delexicalized dialogue history $Z_{del}$, and (domain,slot) pairs $Z_{ds}$, the contextual signals are learned and passed into each $z_{ds}$ vector through a sequence of attention layers. We adopt the multi-head attention mechanism [Vaswani et al., 2017] to project the representations into multiple sub-spaces. The attention mechanism is defined as scaled dot-product attention between query $Q$, key $K$, and value $V$:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}V)$$

Each multi-head attention is followed by a position-wise feed-forward network. The feed-forward is applied to each position separately and identically. We use two linear layers with a ReLU activation in between. The fertility decoder consists of 3 attention layers, each of which learns relevant contextual signals and incorporates them into $z_{ds}$ vectors as input to the next attention layer:

$$Z_{ds}^{out} = \text{Attention}(Z_{ds}, Z_{ds}, Z_{ds}) \in \mathbb{R}^{N \times d}$$

$$Z_{ds}^{out} = \text{Attention}(Z_{ds}^{out}, Z_{del}, Z_{del}) \in \mathbb{R}^{N \times d}$$

$$Z_{ds}^{out} = \text{Attention}(Z_{ds}^{out}, Z, Z) \in \mathbb{R}^{N \times d}$$
We optimize all parameters by jointly training to minimize the weighted sum of the three losses:

\[
L = L_{\text{state}} + \alpha L_{\text{gate}} + \beta L_{\text{fert}}
\]

where \(\alpha \geq 0\) and \(\beta \geq 0\) are hyper-parameters.

For simplicity, we do not express the multi-head and feed-forward equations. We advise the reader to review Transformer network (Vaswani et al., 2017) for more detailed description. The multi-head structure has shown to obtain good performance in many NLP tasks such as NMT (Vaswani et al., 2017) and QA (Dehghani et al., 2019). By adopting this attention mechanism, we allow the models to explicitly obtain signals of potential dependencies across (domain, slot) pairs in the first attention layer, and contextual dependencies in the subsequent attention layers. Adding the delexicalized dialogue history as input can provide important contextual signals as the models can learn the mapping between real-value tokens and generalized domain-slot tokens. To further improve the model capability to capture these dependencies, we repeat the attention sequence for \(T_{\text{fert}}\) times with \(Z_{ds}\). In an attention step \(t\), the output from the previous attention layer \(t-1\) is used as input to current layer to compute \(Z_{ds}^{t}\). The output in the last attention layer \(Z_{ds}^{T_{\text{fert}}}\) is passed to two independent linear transformations to predict fertilities and gates:

\[
P_{\text{gate}} = \text{softmax}(W_{\text{gate}}Z_{ds}^{T_{\text{fert}}})
\]

\[
P_{\text{fert}} = \text{softmax}(W_{\text{fert}}Z_{ds}^{T_{\text{fert}}})
\]

where \(W_{\text{gate}} \in \mathbb{R}^{d \times 3}\) and \(W_{\text{fert}} \in \mathbb{R}^{d \times 10}\). We use the standard cross-entropy loss to train the prediction of gates and fertilities:

\[
L_{\text{gate}} = \sum_{d_g, s_h} - \log(P_{\text{gate}}(Y_{g}^{d_g, s_h}))
\]

\[
L_{\text{fert}} = \sum_{d_f, s_h} - \log(P_{\text{fert}}(Y_{f}^{d_f, s_h}))
\]

### 3.3 State Decoder

Given the generated gates and fertilities, we form the input sequence \(X_{ds \times \text{fert}}\). We filter out any (domain, slot) pairs that have gate either as “none” or “dontcare”. Given the encoded input \(Z_{ds \times \text{fert}}\), we apply a similar attention sequence as used in the fertility decoder to incorporate contextual signals into each \(Z_{ds \times \text{fert}}\) vector. The dependencies are captured at the token level in this decoding stage rather than at domain/slot higher level as in the fertility decoder. After repeating the attention sequence for \(T_{\text{state}}\) times, the final output \(Z_{ds \times \text{fert}}^{T_{\text{state}}}\) is used to predict the state in the following:

\[
P_{\text{state}} = \text{softmax}(W_{\text{state}}Z_{ds \times \text{fert}}^{T_{\text{state}}})
\]

where \(W_{\text{state}} \in \mathbb{R}^{d \times ||V||}\) with \(V\) as the set of output vocabulary. As open-vocabulary DST models do not assume a known slot ontology, our models can generate the candidates from the dialogue history itself. To address OOV problem during inference, we incorporate a pointer network (Vinyals et al., 2015) into the Transformer decoder. We apply dot-product attention between the state decoder output and the stored memory of encoded dialogue history \(Z\):

\[
P_{\text{ptr}} = \text{softmax}(Z_{ds \times \text{fert}}^{T_{\text{state}}}Z^{T})
\]

The final probability of predicted state is defined as the weighted sum of the two probabilities:

\[
P_{\text{state}} = P_{\text{state}} \cdot P_{\text{state}} + (1 - P_{\text{state}}) \cdot P_{\text{ptr}}
\]

\[
P_{\text{gen}} = \text{sigmoid}(W_{\text{gen}}V_{\text{gen}})
\]

\[
V_{\text{gen}} = Z_{ds \times \text{fert}} \oplus Z_{ds \times \text{fert}}^{T_{\text{state}}} \oplus Z_{exp}
\]

where \(W_{\text{gen}} \in \mathbb{R}^{3d \times 1}\) and \(Z_{exp}\) is the expanded vector of \(Z\) to match dimensions of \(Z_{ds \times \text{fert}}\). The final probability is used to train the state generation following the cross-entropy loss function:

\[
L_{\text{state}} = \sum_{d_g, s_h} \sum_{m=0}^{T_{\text{state}}} - \log(P_{\text{state}}(Y_{m}^{d_g, s_h}))
\]

### 3.4 Optimization

We optimize all parameters by jointly training to minimize the weighted sum of the three losses:

\[
L = L_{\text{state}} + \alpha L_{\text{gate}} + \beta L_{\text{fert}}
\]
4 Experiments

4.1 Dataset

MultiWOZ (Budzianowski et al., 2018) is one of the largest publicly available multi-domain task-oriented dialogue dataset with dialogue domains extended over 7 domains. In this paper, we use the new version of the MultiWOZ dataset published by Eric et al. (2019). The new version includes some correction on dialogue state annotation with more than 40% change across dialogue turns. On average, each dialogue has more than one domain. We pre-processed the dialogues by tokenizing, lower-casing, and delexicalizing all system responses following the pre-processing scripts from (Wu et al., 2019). We identify a total of 35 (domain, slot) pairs. Other details of data pre-processing procedures, corpus statistics, and list of (domain, slot) pairs are described in Appendix A.1.

4.2 Training Procedure

We use label smoothing (Szegedy et al., 2016) to train the prediction of dialogue state $Y$ but not for prediction of fertilities $Y_{fert}$ and gates $Y_{gate}$. During training, we adopt 100% teacher-forcing learning strategy by using the ground-truth of $X_{ds \times fert}$ as input to the state decoder. We also apply the same strategy to obtain delexicalized dialogue history i.e. dialogue history is delexicalized from the ground-truth belief state in previous dialogue turn rather than relying on the predicted belief state. During inference, we follow a similar strategy as (Lei et al., 2018) by generating dialogue state turn-by-turn and use the predicted belief state in turn $t-1$ to delexicalize dialogue history in turn $t$. During inference, $X_{ds \times fert}$ is also constructed by prediction $Y_{gate}$ and $Y_{fert}$. We adopt the Adam optimizer (Kingma & Ba, 2015) and the learning rate strategy similarly as (Vaswani et al., 2017). Best models are selected based on the best average joint accuracy of dialogue state prediction in the validation set. All parameters are randomly initialized with uniform distribution (Glorot & Bengio, 2010). We did not utilize any pretrained word- or character-based embedding weights. We tuned the hyper-parameters with grid-search over the validation set (Refer to Appendix A.2 for further details). We implemented our models using PyTorch (Paszke et al., 2017) and released the code on GitHub.*

4.3 Baselines

The DST baselines can be divided into 2 groups: open-vocabulary approach and fixed-vocabulary approach as mentioned in Section 2. Fixed-vocabulary has the advantage of access to the known candidate set of each slot and has a high performance of prediction within this candidate set. However, during inference, the approach suffers from unseen slot values for slots with evolving candidates such as entity names and time- and location-related slots.

4.3.1 Fixed-vocabulary

**GLAD** (Zhong et al., 2018). GLAD uses multiple self-attentive RNNs to learn a global tracker for shared parameters among slots and a local tracker for individual slot. The model utilizes previous system actions as input. The output is used to compute semantic similarity with ontology terms.

**GCE** (Nouri & Hosseini-Asl, 2018). GCE is a simplified and faster version of GLAD. The model removes slot-specific RNNs while maintaining competitive DST performance.

**MDBT** (Ramadan et al., 2018). MDBT model includes separate encoding modules for system utterances, user utterances, and (slot, value) pairs. Similar to GLAD, The model is trained based on the semantic similarity between utterances and ontology terms.

**FJST** and **HJST** (Eric et al., 2019). FJST refers to Flat Joint State Tracker, which consists of a dialog history encoder as a bidirectional LSTM network. The model also includes separate feedforward networks to encode hidden states of individual state slots. HJST follows a similar architecture but uses a hierarchical LSTM network (Serban et al., 2016) to encode the dialogue history.

**SUMBT** (Lee et al., 2019). SUMBT refers to Slot-independent Belief Tracker, consisting of a multi-head attention layer with query vector as a representation of a (domain, slot) pair and key and

*<https://github.com/henryhungle/NADST>
value vector as BERT-encoded dialogue history. The model follows a non-parametric approach as it is trained to minimize a score such as Euclidean distance between predicted and target slots. Our approach is different from SUMBT as we include attention among \textit{(domain, slot)} pairs to explicitly learn dependencies among the pairs. Our models also generate slot values rather than relying on a fixed candidate set.

4.3.2 Open-vocabulary

\textbf{TSCP} [Lei et al., 2018]. TSCP is an end-to-end dialogue model consisting of an RNN encoder and two RNN decoder with a pointer network. We choose this as a baseline because TSCP decodes dialogue state as a single sequence and hence, factor in potential dependencies among slots like our work. We adapt TSCP into multi-domain dialogues and report the performance of only the DST component rather than the end-to-end model. We also reported the performance of TSCP for two cases when the maximum length of dialogue state sequence $L$ in the state decoder is set to 8 or 20 tokens. Different from TSCP, our models dynamically learn the length of each state sequence as the sum of predicted fertilities and hence, do not rely on a fixed value of $L$.

\textbf{DST Reader} [Gao et al., 2019]. DST Reader reformulates the DST task as a reading comprehension task. The prediction of each slot is a span over tokens within the dialogue history. The model follows an attention-based neural network architecture and combines a slot carryover prediction module and slot type prediction module.

\textbf{HyST} [Goel et al., 2019]. HyST model combines both fixed-vocabulary and open-vocabulary approach by separately choosing which approach is more suitable for each slot. For the open-vocabulary approach, the slot candidates are formed as sets of all word n-grams in the dialogue history. The model makes use of encoder modules to encode user utterances and dialogue acts to represent the dialogue context.

\textbf{TRADE} [Wu et al., 2019]. This is the current state-of-the-art model on the MultiWOZ2.0 and 2.1 datasets. TRADE is composed of a dialog history encoder, a slot gating module, and an RNN decoder with a pointer network for state generation. \textbf{SpanPtr} is a related baseline to TRADE as reported by Wu et al. (2019). The model makes use of a pointer network with index-based copying instead of a token-based copying mechanism.

4.4 Results

We evaluate model performance by the joint goal accuracy as commonly used in DST [Henderson et al., 2014b]. The metric compares the predicted dialogue states to the ground truth in each dialogue turn. A prediction is only correct if all the predicted values of all slots exactly match the corresponding ground truth labels. We ran our models for 5 times and reported the average results. For completion, we reported the results in both MultiWOZ 2.0 and 2.1.

As can be seen in Table 2, although our models are designed for non-autoregressive decoding, they can outperform state-of-the-art DST approaches that utilize autoregressive decoding such as [Wu et al., 2019]. Our performance gain can be attributed to the model capability of learning cross-domain and cross-slot signals, directly optimizing towards the evaluation metric of joint goal accuracy rather than just the accuracy of individual slots. Following prior DST work, we reported the model performance on the restaurant domain in MultiWOZ 2.0 in Table 4. In this dialogue domain, our model surpasses other DST models in both Joint Accuracy and Slot Accuracy. Refer to Appendix A.3 for our model performance in other domains in both MultiWOZ2.0 and MultiWOZ2.1.

\textbf{Latency Analysis.} We reported the latency results in term of wall-clock time (in ms) per prediction state of our models and the two baselines TRADE [Wu et al., 2019] and TSCP [Lei et al., 2018] in Table 4. For TSCP, we reported the time cost only for the DST component instead of the end-to-end models. We conducted experiments with 2 cases of TSCP when the maximum output length of dialogue state sequence in the state decoder is set as $L = 8$ and $L = 20$. We varied our models for different values of $T = T_{\text{fert}} = T_{\text{state}} \in \{1, 2, 3\}$. All latency results are reported when running in a single identical GPU. As can be seen in Table 4, NADST obtains the best performance when $T = 3$. The model outperforms the baselines while taking much less time during inference. Our approach is similar to TSCP which also decodes a complete dialogue state sequence rather than individual slots to factor in dependencies among slot values. However, as TSCP models involve sequential
processing in both encoding and decoding, they require much higher latency. TRADE shortens the latency by separating the decoding process among (domain, slot) pairs. However, at the token level, TRADE models follow an auto-regressive process to decode individual slots and hence, result in higher average latency as compared to our approach. In NADST, the model latency is only affected by the number of attention layers in fertility decoder $T_{fert}$ and state decoder $T_{state}$. For approaches with sequential encoding and/or decoding such as TSCP and TRADE, the latency is affected by the length of source sequences (dialog history) and target sequence (dialog state). Refer to Appendix A.3 for visualization of model latency in terms of dialogue history length.

<table>
<thead>
<tr>
<th>Model</th>
<th>MultiWOZ2.1</th>
<th>MultiWOZ2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDBT</td>
<td></td>
<td>15.57%</td>
</tr>
<tr>
<td>SpanPtr</td>
<td></td>
<td>30.28%</td>
</tr>
<tr>
<td>GLAD</td>
<td></td>
<td>35.57%</td>
</tr>
<tr>
<td>GCE</td>
<td></td>
<td>35.57%</td>
</tr>
<tr>
<td>HJST</td>
<td></td>
<td>36.27%</td>
</tr>
<tr>
<td>DST Reader (single)</td>
<td>36.40%</td>
<td>39.41%</td>
</tr>
<tr>
<td>DST Reader (ensemble)</td>
<td>-</td>
<td>42.12%</td>
</tr>
<tr>
<td>TSCP</td>
<td></td>
<td>37.12%</td>
</tr>
<tr>
<td>FJST</td>
<td></td>
<td>38.00%</td>
</tr>
<tr>
<td>HyST (ensemble)</td>
<td>38.10%</td>
<td>44.24%</td>
</tr>
<tr>
<td>SUMBT</td>
<td></td>
<td>45.60%</td>
</tr>
<tr>
<td>TRADE</td>
<td></td>
<td>49.04%</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td></td>
<td><strong>50.52%</strong></td>
</tr>
</tbody>
</table>

Table 2: DST Joint Accuracy metric on MultiWOZ 2.1 and 2.0. †: results reported on MultiWOZ2.0 leaderboard. *: results reported by Eric et al. (2019). Best results are highlighted in bold.

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Acc</th>
<th>Slot Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADE</td>
<td>45.60%</td>
<td>362.15×2.12</td>
</tr>
<tr>
<td>TSCP (L=8)</td>
<td>32.15%</td>
<td>493.44×1.56</td>
</tr>
<tr>
<td>TSCP (L=20)</td>
<td>37.12%</td>
<td>767.57×1.00</td>
</tr>
<tr>
<td><strong>Ours (T=1)</strong></td>
<td>42.98%</td>
<td>15.18×50.56</td>
</tr>
<tr>
<td><strong>Ours (T=2)</strong></td>
<td>45.78%</td>
<td>21.67×35.42</td>
</tr>
<tr>
<td><strong>Ours (T=3)</strong></td>
<td>49.04%</td>
<td>27.31×28.11</td>
</tr>
</tbody>
</table>

Table 3: DST joint accuracy and slot accuracy on MultiWOZ2.0 restaurant domain. Baseline results (except TSCP) were from Wu et al. (2019).

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Acc</th>
<th>Latency</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADE</td>
<td>45.60%</td>
<td>362.15×2.12</td>
<td></td>
</tr>
<tr>
<td>TSCP (L=8)</td>
<td>32.15%</td>
<td>493.44×1.56</td>
<td></td>
</tr>
<tr>
<td>TSCP (L=20)</td>
<td>37.12%</td>
<td>767.57×1.00</td>
<td></td>
</tr>
<tr>
<td><strong>Ours (T=1)</strong></td>
<td>42.98%</td>
<td>15.18×50.56</td>
<td></td>
</tr>
<tr>
<td><strong>Ours (T=2)</strong></td>
<td>45.78%</td>
<td>21.67×35.42</td>
<td></td>
</tr>
<tr>
<td><strong>Ours (T=3)</strong></td>
<td>49.04%</td>
<td>27.31×28.11</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Latency analysis on MultiWOZ2.1. Latency is reported in terms of wall-clock time in ms per prediction state.

**Ablation Analysis.** We conduct an extensive ablation analysis with several variants of our models in Table 5. Besides the results of DST metrics, Joint Slot Accuracy and Slot Accuracy, we reported the performance of the fertility decoder in Joint Gate Accuracy and Joint Fertility Accuracy. These metrics are computed similarly as Joint Slot Accuracy in which the metrics are based on whether all predictions of gates or fertilities match the corresponding ground truth labels. We also reported the Oracle Joint Slot Accuracy and Slot Accuracy when the models are fed with ground truth $X_{ds \times fert}$ and $X_{del}$ labels instead of the model predictions. We noted that the model fails when positional encoding of $X_{ds \times fert}$ is removed before being passed to the state decoder. The performance drop can be explained because $PE$ is responsible for injecting sequential attributes to enable non-autoregressive decoding. Second, we also note a slight drop of performance when slot gating is removed as the models have to learn to predict a fertility of 1 for “none” and “dontcare” slots as well. Third, removing $X_{del}$ as an input reduces the model performance, mostly due to the sharp decrease in Joint Fertility Accuracy. Lastly, removing pointer generation and relying on only $P_{state}^{vocab}$ affects the model performance as the models are not able to infer slot values unseen during training, especially for slots such as restaurant-name and train-arriveby. We conduct other ablation experiments and report additional results in Appendix A.3.
Table 5: Ablation analysis on MultiWOZ 2.1 on 4 components: partially delexicalized dialogue history $X_{del}$, slot gating, positional encoding $PE(X_{ds \times fert})$, and pointer network.

Auto-regressive DST. We conduct experiments that use an auto-regressive state decoder and keep other parts of the model the same. For the fertility decoder, we do not use Equation 14 and 16 as fertility becomes redundant in this case. We still use the output to predict slot gates. Similar to TRADE, we use the summation of embedding vectors of each domain and slot pair as input to the state decoder and generate slot value token by token. First, From Table 5 we note that the performance does not change significantly as compared to the non-autoregressive version. This reveals that our proposed NADST models can predict fertilities reasonably well and performance is comparable with the auto-regressive approach. Second, we observe that the auto-regressive models are less sensitive to the use of system action in dialogue history delexicalization. We expect this as predicting slot gates is easier than predicting fertilities. Finally, we note that our auto-regressive model variants still outperform the existing approaches. This could be due to the high-level dependencies among (domain, slot) pairs learned during the first part of the model to predict slot gates.

Table 6: Performance of auto-regressive model variants on MultiWOZ2.0 and 2.1. Fertility prediction is removed as fertility becomes redundant in auto-regressive models.

Visualization and Qualitative Evaluation. In Figure 4, we include two examples of dialogue state prediction and the corresponding visualization of self-attention scores of $X_{ds \times fert}$ in state decoder. In each heatmap, the highlighted boxes express attention scores among non-symmetrical domain-slot pairs. In the first row, 5 attention heads capture the dependencies of two pairs (train-leaveat, train-arriveby) and (train-departure, train-destination). The model prediction for these two slots matches the gold labels: (train-leaveat, 09:50), (train-arriveby, 11:30) and (train-departure, cambridge), (train-destination, ely) respectively. In the second row, besides slot-level dependency between domain-slot pairs (taxi-departure, taxi-destination), token-level dependency is exhibited through the attention between attraction-type and attraction-name. By attending on token representations of attraction-name with corresponding output “christ college”, the models can infer “attraction-type=college” correctly. In addition, our model also detects contextual dependency between train-departure and attraction-name to predict “train-departure=christ college.” Refer to Appendix A.4 for the dialogue history with gold and prediction states of these two sample dialogues.

5 Conclusion

We proposed NADST, a novel Non-Autoregressive neural architecture for DST that allows the model to explicitly learn dependencies at both slot-level and token-level to improve the joint accuracy rather than just individual slot accuracy. Our approach also enables fast decoding of dialogue states by adopting a parallel decoding strategy in decoding components. Our extensive experiments on the well-known MultiWOZ corpus for large-scale multi-domain dialogue systems benchmark show that our NADST model achieved the state-of-the-art accuracy results for DST tasks, while enjoying a substantially low inference latency which is an order of magnitude lower than the prior work.
REFERENCES


A APPENDIX

A.1 DATASET PRE-PROCESSING

We follow similar data preprocessing procedures as Budzianowski et al. (2018) and Wu et al. (2019) on both MultiWOZ 2.0 and 2.1. The resulting corpus includes 8,438 multi-turn dialogues in training set with an average of 13.5 turns per dialogue. For the test and validation set, each includes 1,000 multi-turn dialogues with an average of 14.7 turns per dialogue. The average number of domains per dialogue is 1.8 for training, validation, and test sets. The MultiWOZ corpus includes much larger ontology than previous DST datasets such as WOZ (Wen et al., 2017) and DSTC2 (Henderson et al., 2014a). We identified a total of 35 (domain, slot) pairs across 7 domains. However, only 5 domains are included in the test data. Refer to Table 7 for the statistics of dialogues in these 5 domains.

<table>
<thead>
<tr>
<th>Domain</th>
<th>attraction</th>
<th>hotel</th>
<th>restaurant</th>
<th>taxi</th>
<th>train</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot</td>
<td>area name type</td>
<td>area bookday bookpeople bookstay internet name parking pricerange stars type</td>
<td>area bookday bookpeople booktime food name pricerange</td>
<td>arriveby departure destination leavet</td>
<td>arriveby bookpeople day departure destination leavet</td>
<td>-</td>
</tr>
<tr>
<td>train</td>
<td>3,381</td>
<td>3,103</td>
<td>2,717</td>
<td>3,813</td>
<td>1,654</td>
<td>8,438</td>
</tr>
<tr>
<td>val</td>
<td>416</td>
<td>484</td>
<td>401</td>
<td>438</td>
<td>207</td>
<td>1,000</td>
</tr>
<tr>
<td>test</td>
<td>394</td>
<td>494</td>
<td>395</td>
<td>437</td>
<td>195</td>
<td>1,000</td>
</tr>
</tbody>
</table>

Table 7: Summary of MultiWOZ dataset 2.1

A.2 MODEL HYPER-PARAMETERS

We employed dropout (Srivastava et al., 2014) of 0.2 at all network layers except the linear layers of generation network components and pointer attention components. We used a batch size of 32, embedding dimension \(d = 256\) in all experiments. We also fixed the number of attention heads to 16 in all attention layers. We shared the embedding weights to embed domain and slot tokens as input to fertility decoder and state decoder. We also shared the embedding weights between dialogue history encoder and state generator. We varied our models for different values of \(T = T_{\text{fert}} = T_{\text{state}} \in \{1, 2, 3\}\). In all experiments, the warmup steps are fine-tuned from a range from 13K to 20K training steps.

A.3 ADDITIONAL RESULTS

**Domain-specific Results.** We conduct experiments to evaluate our model performance in all 5 test domains in MultiWOZ2.0 and 2.1. From Table 8, our models perform better in restaurant and attraction domain in general. The performance in the taxi and hotel domain is significantly lower than other domains. This could be explained as the hotel domain has a complicated slot ontology with 10 different slots, larger than the other domains. For the taxi domain, we observed that dialogues with this domain are usually of multiple domains, including the taxi domain in combination with other domains. Hence, it is more challenging to track dialogue states in the taxi domain.

**Latency Results.** We visualized the model latency against the length of dialogue history in Figure 2 and 3. In Figure 2 we only plot with dialogue history length up to 80 tokens as TSCP models do not use the full dialogue history as input. In Figure 3 for a fair comparison between TRADE and NADST, we plot the latency of the original TRADE which decodes dialogue state slot by slot and a new version of TRADE* model which decodes individual slots following a parallel decoding mechanism. Since TRADE independently generates dialogue state slot by slot, we enable parallel generation simply by feeding all slots into models at once (without impacts on performance). However, at the token level, TRADE* still follows an autoregressive decoding framework. Compared to TRADE* and TSCP, our model latency is only dependent on the model complexity i.e. the number
of attention layers $T = T_{fert} = T_{state}$. For TRADE$^*$ and TSCP, the model latency increases as dialogue extends over time while NADST latency is almost constant. The non-constant latency is mostly due to overhead processing such as delexicalizing dialogue history. Our approach is, hence, suitable especially for dialogues in multiple domains as they usually extend over more number of turns (e.g. 13 to 14 turns per dialogue in average in MultiWOZ corpus) In Figure [3] we noted that the latency of the original TRADE is almost unchanged as the dialogue history extends. This is most likely due to the model having to decode all possible (domain, slot) pairs rather than just relevant pairs as in NADST and TSCP. The TRADE$^*$ shows a clearer increasing trend of latency because the parallel process is independent of the number of (domain,slot) pairs considered. TRADE$^*$ still requires more time to decode than NADST as we also parallelize decoding at the token level.

Table 8: Additional domain-specific results of our model in MultiWOZ2.0 and MultiWOZ2.1. The model performs best with the restaurant domain and worst with the taxi domain.

<table>
<thead>
<tr>
<th>Domain</th>
<th>MultiWOZ2.1</th>
<th>MultiWOZ2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Joint Acc</td>
<td>Slot</td>
</tr>
<tr>
<td>Hotel</td>
<td>48.76%</td>
<td>97.70%</td>
</tr>
<tr>
<td>Train</td>
<td>62.36%</td>
<td>98.36%</td>
</tr>
<tr>
<td>Attraction</td>
<td>66.83%</td>
<td>98.89%</td>
</tr>
<tr>
<td>Restaurant</td>
<td>65.37%</td>
<td>98.78%</td>
</tr>
<tr>
<td>Taxi</td>
<td>33.80%</td>
<td>96.69%</td>
</tr>
</tbody>
</table>

Figure 2: Comparison of model latency as wall-clock time (in ms) per prediction of complete dialogue state (not by individual slot). The latency is plotted against the length of the dialogue history. We compare our models with TSCP (Lei et al., 2018) with varied maximum output length of dialogue states $L = 8$ and $L = 20$. We vary our models with different values of number of attention layers $T = T_{fert} = T_{state} = 1, 2, 3$. Our models are more scalable as the latency does not change significantly when dialogue history extends over time.

Ablation Results. We conduct additional ablation experiments by varying the proportion of prediction values vs. ground-truth values for $X_{del}$ and $X_{dsxfert}$ as input to the models. As can be seen in Table [9] the model performance increases gradually as the proportion of prediction input %pred reduces from 100% (true prediction) to 0% (oracle prediction). In particular, we observe more sig-
Figure 3: Comparison of model latency as wall-clock time (in ms) per prediction of complete dialogue state. The latency is plotted against the length of the dialogue history. For a fair comparison, we compare our models with TRADE ([Wu et al., 2019]) in 2 cases: original TRADE which decodes dialogue state slot by slot and TRADE∗ which decodes dialogue state in parallel at slot-level. Here we plot the base-10 logarithm of latency to show the difference between the 2 cases of TRADE. We vary our models with different values of number of attention layers $T = T_{fert} = T_{state} = 1, 2, 3$. Our models are more scalable as the latency does not change significantly when dialogue history extends over time.

Table 9: Additional results of our model in MultiWOZ2.1 when we assume access to the ground-truth labels of $X_{del}$ and $X_{ds \times fert}$ (oracle prediction). We vary the the percentage of using the model prediction $\hat{X}_{del}$ and $\hat{X}_{ds \times fert}$ from 100% (true prediction) to 0% (oracle prediction).
Figure 4: Heatmap visualization of self-attention scores of 5 heads between $Z_{ds \times fert}$ representations in the state decoder. The corresponding prediction output for each representation is presented on the right side. The examples are for the $6^{th}$ turn in dialogue ID MUL0536 (upper row) and PMUL3759 (lower row) in MultiWOZ2.1.

A.4 Sample Prediction Output

We extracted prediction output in all turns for 2 example dialogues: MUL0536 and PMUL3759.
Turn 1  
**Human:** i am looking for abbey pool and astroturf pitch can you help me ?  
**Gold Dialog State:** (attraction-name, abbey pool and astroturf pitch)  
**Predicted Dialog State:** (attraction-name, abbey pool and astroturf pitch)

**Turn 2**  
**System:** yes , abbey pool and astroturf pitch is a swimmingpool east of town . their number is 01223902088 , and address is pool way , whitehill road , off newmarket road . postcode cb58nt .  
**Human:** thank you very much for the information . that is all i needed help with . have a nice day .  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch)

**Turn 3**  
**System:** you are welcome . let me know if i can do anything else for you .  
**Human:** i actually do need to find a train going to ely .  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool), (train-destination-ely)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-destination-ely)

**Turn 4**  
**System:** is that leaving from cambridge ? and if so , what time would you like to arrive in ely ?  
**Human:** yes , i will be leaving cambridge and going to ely . i would like it to arrive by 11:30 .  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool), (train-arriveby, 11:30), (train-departure, cambridge), (train-destination, ely)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-arriveby, 11:30), (train-departure, cambridge), (train-destination, ely)

**Turn 5**  
**System:** what day would you be traveling ? there are 2,828 trains on that route . there is a train that departs friday at 9:50 and will arrive in ely at 10:07 .  
**Human:** oops ! i guess forgot to mention it ' s thursday that i need to travel .  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely)

**Turn 6**  
**System:** there are 3 trains that would fit , leaving at 05:50 , 07:50 , or 09:50 . can i get info for the 09:50 the price and the trains id please ?  
**Human:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely)  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-arriveby, 11:30), (train-departure, cambridge), (train-destination, ely), (train-leaveat, 09:50)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-arriveby, 11:30), (train-departure,cambridge), (train-destination, ely), (train-leaveat, 09:50)

**Turn 7**  
**System:** certainly . the train s id is tr1923 , and the price for a ticket is 4.40 pounds .  
**Human:** great , thank you ! that will be all i need for now .  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely), (train-leaveat, 09:50)

**Turn 8**  
**System:** are you certain you do not need further assistance ?  
**Human:** 9:50 departure , 4.40 pounds , tr1923 . i got it , thank you !  
**Gold Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (attraction-type, swimming pool), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely)  
**Predicted Dialog State:** (attraction-area, east), (attraction-name, abbey pool and astroturf pitch), (train-arriveby, 11:30), (train-departure, thursday), (train-destination, cambridge), (train-destination, ely), (train-leaveat, 09:50)

Table 10: Full set of predicted dialogue states for dialogue ID MUL0536 in MultiWOZ2.1.
Figure 5: Visualization of all attention heads in the last attention step $T_{state}$ in the state decoder. The DST prediction is done for the 6th turn in dialogue ID MUL0536 in MultiWOZ2.1.
<table>
<thead>
<tr>
<th>Turn</th>
<th>Human</th>
<th>Gold Dialog State</th>
<th>Predicted Dialog State</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td>what’s your favorite college to visit in Cambridge?</td>
<td>attraction-type, college</td>
<td>attraction-type, college</td>
</tr>
<tr>
<td><strong>2</strong></td>
<td>i like Christ’s college in the center of town, but there are many others throughout the city. Which part of the city did you want to visit?</td>
<td>attraction-name, Christ college, attraction-type, college</td>
<td>attraction-name, Christ college, attraction-type, college</td>
</tr>
<tr>
<td><strong>3</strong></td>
<td>the area is centre and the phone number is 01223334900. Is there anything else I can help you with?</td>
<td>attraction-name, Christ college, attraction-type, college, (restaurant-food, modern american), (restaurant-pricerange, moderate)</td>
<td>attraction-name, Christ college, attraction-type, college, (restaurant-food, modern american), (restaurant-pricerange, moderate)</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>i am sorry, there’s no restaurant serving specifically North American or American food in my database. Is there another type of food you would consider?</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate)</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate)</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td>there are 3 modern European restaurants in the center and 1 in the south. Do you have a preference?</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td>de Luca Cucina and Bar’s phone number is 01223356666. Could you help me get a taxi to get from the college to the restaurant?</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
</tr>
<tr>
<td><strong>7</strong></td>
<td>what time would you like to leave the college? I can book you a taxi to take you to the restaurant if you would like.</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
</tr>
<tr>
<td><strong>8</strong></td>
<td>i have booked you a taxi leaving at 12:45. The car will be a red Toyota and contact number is 07350032543. Anything else today?</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
</tr>
<tr>
<td><strong>9</strong></td>
<td>will you need anymore information concerning your stay? That is all. Thank you very much.</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
<td>attraction-name, Christ college, (restaurant-food, modern European), (restaurant-pricerange, moderate), (taxi-departure, Christ college), (taxi-destination, de Luca Cucina and Bar)</td>
</tr>
</tbody>
</table>

Table 11: Full set of predicted dialogue states for dialogue ID PMUL3759 in MultiWOZ2.1.
Figure 6: Visualization of all attention heads in the last attention step $T_{state}$ in the state decoder. The DST prediction is done for the 6\textsuperscript{th} turn in dialogue ID PMUL3759 in MultiWOZ2.1.