Knowledge-Aware Graph-Enhanced GPT-2 for Dialogue State Tracking

Weizhe Lin   Bo-Hsiang Tseng   Bill Byrne
Department of Engineering, University of Cambridge, United Kingdom
wl356@cam.ac.uk   bht26@cam.ac.uk   bill.byrne@eng.cam.ac.uk

Abstract

Dialogue State Tracking is central to multi-domain task-oriented dialogue systems, responsible for extracting information from user utterances. We present a novel hybrid architecture that augments GPT-2 with representations derived from Graph Attention Networks in such a way to allow causal, sequential prediction of slot values. The model architecture captures inter-slot relationships and dependencies across domains that otherwise can be lost in sequential prediction. We report improvements in state tracking performance in MultiWOZ 2.0 against a strong GPT-2 baseline and investigate a simplified sparse training scenario in which DST models are trained only on session-level annotations but evaluated at the turn level. We further report detailed analyses to demonstrate the effectiveness of graph models in DST by showing that the proposed graph modules capture inter-slot dependencies and improve the predictions of values that are common to multiple domains.

1 Introduction

This paper investigates two aspects of dialogue state tracking (DST) for multi-domain task-oriented dialogue (Budzianowski et al., 2018). We present a novel hybrid architecture that augments GPT-2 (Radford et al., 2019) with dialogue act representations derived from Graph Attention Networks (GATs) (Veličković et al., 2018) in such a way that allows causal, sequential prediction of slot values while explicitly modelling the relationships between slots and values across domains. Our approach uses GATs to improve predictions of values that are shared across domain-slots and that might otherwise be treated independently. As a related line of work, we investigate a form of sparsely supervised DST training and find that our hybrid architecture offers improved robustness with weak supervision.

DST can be improved by modelling the relationship between slots and values across domains. This has been explored recently by Zhou and Small (2019) who suggest three types of relationships between domain-slots pairs that can be modelled explicitly: (1) pairs that share the same candidate set, such as `<restaurant-bookday>` and `<hotel-bookday>`; (2) pairs whose candidate values are subsets, as could happen with `<restaurant-name>` and `<taxi-destination>` if the candidate set of the first belongs to that of the second; and (3) correlated values between domain-slot pairs, such as when the 'star' level of a booked hotel correlates with the price range of a reserved restaurant.

Graph Neural Networks (GNNs) have been proposed to captures the interactions among slots and values and to improve DST performance (Zhou and Small, 2019; Chen et al., 2020; Wu et al., 2020). These relationships can be represented as edges in graph-based models, where domains, slots, and values are nodes in the graphs. However previous work has not explored quantitatively or in depth how graph models utilize the relationships they model. Chen et al. (2020) and Wu et al. (2020) provide example cases where the predictions of correlated values were potentially enhanced by their model, while Zhou and Small (2019) and Zhu et al. (2020) present ablation studies showing marginal improvements brought by their graph modules. Zhu et al. (2020) and Wu et al. (2020) further show joint accuracies over different dialogue turns, but there is more that can be said about how GATs can improve DST. One of the aims of this paper is to more deeply analyze how graph models can lead to improved DST on top of an already good GPT-2 baseline system.

Graph models may also compensate for some potential drawbacks associated with using generative models for DST. As a well-known generative model, GPT-2 offers powerful, left-to-right

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7871–7881
November 7–11, 2021.

generation incorporating a causal attention mechanism. We note that Hosseini-Asl et al. (2020) have demonstrated that GPT-2 can identify slot values as a prediction task, with variable length token sequences produced sequentially with interspersed special tokens indicating slot boundaries. The ability to easily generate token sequences of arbitrary lengths is a valuable feature of the model, although it may come at the expense of modelling power relative to models with non-causal attention mechanisms, such as BERT (Devlin et al., 2019; Shan et al., 2020). In particular, GPT-2's causality requires that the prediction of later slot values can depend explicitly on previously predicted slot values, but that the reverse is not possible. This can lead to decreased performance in predicting slot values that occur early on. We find that augmenting GPT-2 prediction with representations derived from GATs allows some sharing of information between slots prior to prediction to improve this GPT-2 limitation.

Capturing the relationships of slot values across domains also offers the opportunity to make better use of limited training data, particularly in sparsely supervised and weakly supervised scenarios (Liang et al., 2021). In a 'Last Turn' annotation scenario, annotations are available only for the final turn of a task-oriented dialogue. This is unlike the fully-annotated MultiWOZ setting, which offers turn-level annotations throughout the entire dialogue session. As an annotation option, generating summary annotations at the completion of a recorded session is an attractive alternative to creating a detailed, turn-by-turn annotation of the entire dialogue (Liang et al., 2021). If it is possible to use only these session-level annotations to train a DST system that still achieves acceptable tracking performance, the chore of creating new annotated DST datasets could be made much easier. The challenges in using this summary data are significant, however. Using only the final-turn annotations in MultiWOZ 2.0 reduces the training set to 14.3% of its original size (in annotated turns).

We summarize the contributions of our work as follows:

(1) We propose a novel hybrid architecture that integrates GPT-2 with Graph Attention Networks (GATs) for dialogue state tracking. The model is shown to be robust when training samples are significantly reduced under sparse supervision.

(2) We demonstrate that our architecture also mitigates a limitation of DSTs based on GPT-2 alone, associated with generating domain-slot values in a Left-to-Right manner.

(3) We investigate how knowledge-aware models capture relationships between domain-slots and show how using graphs can improve prediction of inter-dependent slot values.

While we do show DST accuracy improvements over a strong GPT-2 baseline, we emphasise that our aim is mainly to investigate and improve prediction of domain-slot values using relationships that otherwise are left unmodelled by the baseline.

2 Related Work

Statistical DST prioritises general and extensible systems based on machine-learning architectures (Wu et al., 2019; Zhang et al., 2019; Huang et al., 2020; Lee et al., 2020). Systems must be able to predict slot values from domain-specific lists such as list of hotel names as well as from more open-ended categories such as days, prices, and times. Recent trends are to combine several strategies to deal differently with the two types of values (Zhang et al., 2019; Zhou and Small, 2019; Heck et al., 2020). For example, Zhang et al. (2019) combine a span predictor for non-enumerable slot values and a cosine similarity matching that exploits a BERT model to extract representations for enumerable slot values, with a dual-strategy model jointly handling both types of slot values; Zhou and Small (2019) use both a span predictor and a candidate classifier and combine their predictions with gating functions. Our work is based on GPT-2 and we note that generative models such as GPT-2 are less widely used in DST tasks, possibly because they raise additional challenges for information aggregation subject to the causality, as discussed in Sec. 1. However these recent results show that these models can yield competitive DST accuracy (Hosseini-Asl et al., 2020; Yang et al., 2021) .

Previous work has addressed sharing information between slots either by explicitly copying values (Ouyang et al., 2020; Heck et al., 2020) or by sharing embeddings (Hu et al., 2020; Zhou and Small, 2019; Chen et al., 2020). Beyond copying and sharing, as we note in Sec. 1 Zhou and Small (2019) developed a graph attention network, and Chen et al., 2020 also developed a schema-guided multi-domain approach embedding slot relations in edges of graph neural networks. Zhu et al. (2020) enhanced a strong base model SOM-DST (Kim et al., 2020) with a schema graph to exploit relations among domain-slots. GCDST (Wu et al., 2020) uses a state graph to transfer domain-slot features and hard-copy states directly from historical states.

3 Graph Neural Networks

In this section we review Graph Attention Networks (GATs) (Veličković et al., 2018; Li et al., 2021) as will be used in this paper.

A weighted undirected graph at each dialogue turn t is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node set $\mathcal{V}$ consisting of N nodes $\{v_i\}$, and an edge set $\mathcal{E}$ containing all edges between nodes. We define an $N \times N$ binary symmetric adjacency matrix $\mathbf{S}$, where $[\mathbf{S}]_{ij} = 0$ if $(v_i, v_j) \notin \mathcal{E}$ and 1 otherwise. Associated with each node v_i are feature vectors $\mathbf{x}^i \in \mathbf{R}^F$. These are gathered into matrices $\mathbf{X}$ of dimension $N \times F$, where F is the input feature size.

Note that $\mathbf{SX}$ is mathematically equivalent to passing the features of each graph node to its neighbours. In this way $\mathbf{S}^k\mathbf{X} = \mathbf{S}(\mathbf{S}^{k-1}\mathbf{X})$ is equivalent to k rounds of feature exchanges with neighbours. As illustrated in Fig. 1, $k = 0$ is self-connection, while $k > 0$ aggregates features from k nodes away.

A GAT layer transforms an input $\mathbf{X} \in \mathbf{R}^{N\times F}$ to an output $\mathcal{G}(\mathbf{X}) \in \mathbf{R}^{N\times G}$ as follows. Each K-hop GAT layer consists of P attention heads $\mathcal{A}^{(p)}$ which incorporate $k = 0, ..., K-1$ rounds of feature aggregation (as shown in Fig. 1) across the graph as

$$\begin{aligned} \mathcal{A}^{(p)}(\mathbf{X};\mathbf{S}) &= \sum_{k=0}^{K-1} (\mathbf{E} \odot \mathbf{S})^k \mathbf{X} \mathbf{A}_k^{(p)} \\ \mathcal{G}(\mathbf{X}) &= \frac{1}{P} \sum_{p=1}^{P} \sigma\big[\mathcal{A}^{(p)}(\mathbf{X};\mathbf{S})\big], \end{aligned} \tag{1}$$

where the $\{\mathbf{A}_k^{(p)}\}_{k=0}^{K-1}$ are $\mathbf{R}^{F\times G}$ linear feature transforms and $\sigma(.)$ is a non-linear activation function. The values of the $N \times N$ attention matrix $\mathbf{E}$ are computed over $\mathbf{X}$ as

$$\begin{aligned} [\mathbf{E}]_{ij} &= \frac{\exp\left(LeakyReLU(e_{ij})\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(LeakyReLU(e_{ik})\right)} \\ e_{ij} &= (\mathbf{x}^i)^\top \boldsymbol{Q}^{(p)} \mathbf{x}^j, \end{aligned} \tag{2}$$

where $\mathcal{N}_i$ are the neighbouring nodes of node v_i, and $\boldsymbol{Q}^{(p)}$ are trainable $F \times F$ matrices used in computing attention. In this way a GAT layer aggregates features selectively by assigning dynamic weights to graph edges based on the input node features.

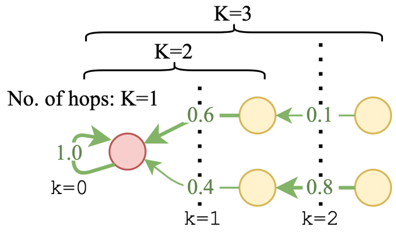


Figure 1: Illustration of GATs. $k = 0$ is self-connection, and $k \geq 1$ passes the features of other nodes to the node being evaluated. The values on the links are attention values, which weight the passing features.

GATs are formed as a cascade of L GAT layers $\mathcal{G}_\ell$, each with its own multi-headed graph attention mechanisms $\mathcal{A}_\ell^p$. At time t, the GAT transforms a set of input features $\mathbf{X}_t^{(0)}$ to a set of output features $\mathbf{X}_t^{(L)}$ as

$$\mathbf{X}_t^{(\ell)} = \mathcal{G}_\ell(\mathbf{X}_t^{(\ell-1)}) \text{ for } \ell = 1, \ldots, L \tag{3}$$

Note that in this paper, we set output dimension $G = F$ for all GAT layers such that the GAT output features have the same dimensions as the input.

4 Dialogue State Tracking with GPT-2 and Graph Neural Networks

We take a three-step approach to incorporating GNNs into GPT-2 for dialogue state tracking (see Fig 2). At each turn we first present GPT-2 with the dialogue history to generate features for all possible domain-slots and values in the ontology. These features are then fed into a GAT which captures relationships amongst domain-slots and values. The features produced at the output layer of the GAT are then incorporated into a second application of GPT-2 which performs the actual prediction of the dialogue state values.

4.1 Domain-Slot and Value Embeddings

The first step is to extract features of both domain-slots and values in the ontology. The dialogue history H_t at turn t is a concatenation of user utterances and system responses, separated with special tokens: H_t = 'u_t `<SYS>` s_{t-1} `<USR>` u_{t-1} ... `<SYS>` s_1 `<USR>` u_1'. From the ontology, we construct a string for all domain-slots as follows: F = '*hotel name* `<hotel-name>` *taxi departure* `<taxi-departure>`...' . The string F contains all domain-slots in the ontology and does

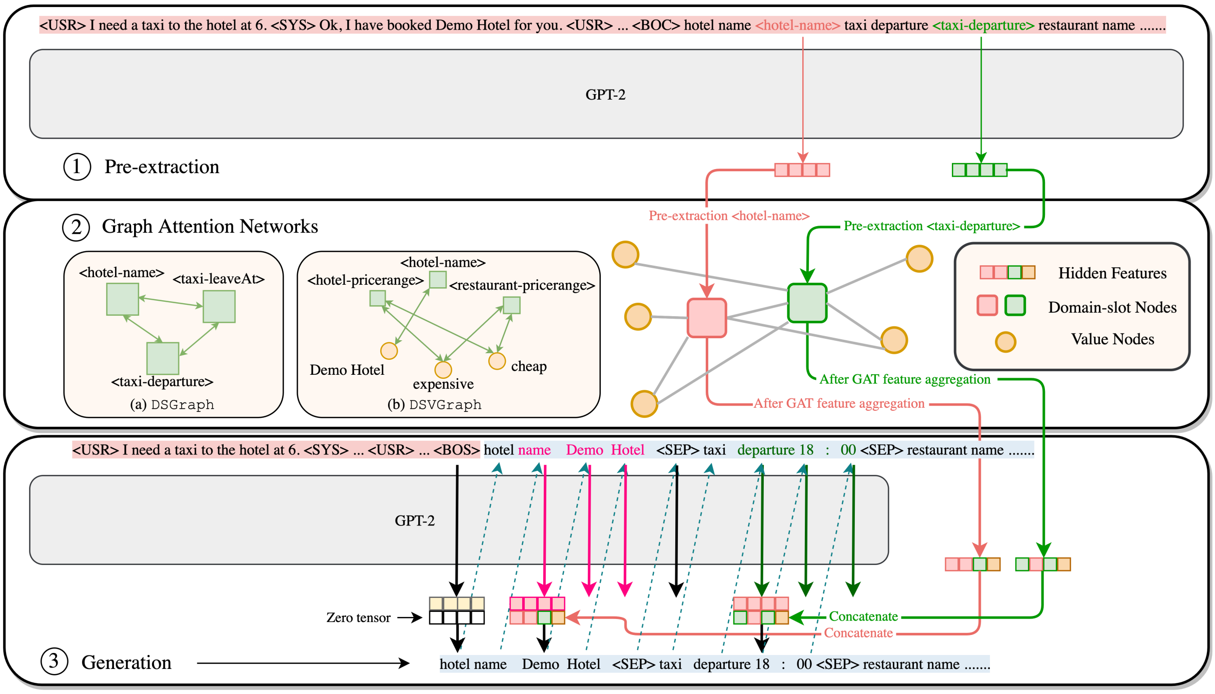

<USR> I need a taxi to the hotel at 6. <SYS> Ok, I have booked Demo Hotel for you. <USR> ... <BOC> hotel name <hotel-name> taxi departure <taxi-departure> restaurant name ....
GPT-2
① Pre-extraction
② Graph Attention Networks
<hotel-name>
<taxi-leaveAt>
<taxi-departure>
(a) DSGraph
<hotel-name>
<hotel-pricerange>
<restaurant-pricerange>
Demo Hotel
expensive
cheap
(b) DSVGraph
Pre-extraction <hotel-name>
Pre-extraction <taxi-departure>
After GAT feature aggregation
After GAT feature aggregation
Hidden Features
Domain-slot Nodes
Value Nodes
<USR> I need a taxi to the hotel at 6. <SYS> ... <USR> ... <BOS> hotel name Demo Hotel <SEP> taxi departure 18 : 00 <SEP> restaurant name ....
GPT-2
Zero tensor
Concatenate
Concatenate
③ Generation
hotel name Demo Hotel <SEP> taxi departure 18 : 00 <SEP> restaurant name ....

Figure 2: The workflow of the proposed model: ① The model extracts domain-slot embeddings from dialogue history, without knowing the ground truth; ② domain-slot embeddings are passed into Graph Attention Networks for feature aggregation and information exchanges; ②(a)-②(b) two types of graph connectivity used in our experiments; ③ the updated domain-slot features are fed into the causal generation process of corresponding slots. Tokens shaded with red are model inputs, while tokens shaded with blue are generation outputs. For better visualization, only two domain-slot pairs are presented (`<hotel-name>` and `<taxi-departure>`).

not change with samples. The domain-slots appear in a fixed order and each is preceded by a brief text description to provide context to GPT-2 in producing features.

To produce domain-slot features at dialogue turn t, the string 'H_t *<BOC>* F' is presented to GPT-2. Since the domain-slots are fixed and appear in a prescribed order in F, there is a straightforward link between the positions of domain-slots in the input and their embeddings in the GPT-2 output layer. For example, the feature for `<taxi-depature>` can be found in the same position of the output embedding sequence as that domain-slot appears in the input, as shown by arrows in Fig.2 ①: Pre-extraction.

To produce embeddings for all possible values in the ontology at turn t the embedding layer of the GPT-2 is used. Therefore, this representation is fixed from turn to turn until the embedding layer is updated in back propagation. Some values may consist of multiple tokens, e.g. *'Demo Hotel'* for the domain-slot `<hotel-name>`. A single vector for each multi-token value is found by averaging the features of each token.

At dialogue turn t, the domain-slot features and the value features are gathered into matrices $X_t^s \in \mathbf{R}^{N_s \times h}$ and $X_t^v \in \mathbf{R}^{N_v \times h}$, where there are N_v values and N_s domain-slots, and h is the size of the hidden layer of the GPT-2 Transformer.

4.2 Inter-slot Information Exchange

We will use two types of GATs: `DSGraph` and `DSVGraph`. In `DSGraph`, there are N_s nodes, each representing a domain-slot pair. All nodes are connected to each other to allow nodes to exchange features as shown in Fig.2 ② (a). In `DSVGraph`, there are N_s domain-slot nodes and N_v value nodes, each of the latter representing a possible value. If a value is in the candidate set of a domain-slot pair, then the corresponding value node and domain-slot node are connected, as shown in Fig.2 ② (b). The domain-slot nodes are not otherwise connected.

With features for domain-slots and values extracted in Sec. 4.1, we use GATs to transform the features to capture the relationships between domain-slots and values. The inputs to the GATs are

$$\mathbf{X}_t^{(0)} = \begin{cases} X_t^s \in \mathbf{R}^{N_s \times h} & \textit{in}\ \texttt{DSGraph}, \\ X_t^s || X_t^v \in \mathbf{R}^{(N_s+N_v) \times h} & \textit{in}\ \texttt{DSVGraph} \end{cases}.$$

We use only the resulting domain-slot embeddings after graph operations, and thus we extract the first N_s items of the output tensor $\mathbf{X}_t^{(L)}$ and gather them into a matrix $\mathbf{G}_t \in \mathbf{R}^{N_s \times h}$.

4.3 Dialog State Prediction

Finally, we present the string 'H_t <BOS>' to the GPT-2 model to predict the dialogue state. The model is required to generate output Y_t, a sequence of tokens of serialized domain-slot pairs and corresponding values: $Y_t =$ '*hotel name Demo Hotel* <SEP> *taxi departure 18 : 00* <SEP> ... <EOS>'. The model is trained to generate the name of each domain-slot, its predicted value, and finally a separation token <SEP> before proceeding to the prediction of the next domain-slot. Note that the value 'none' is generated for empty/not mentioned domain-slot values and thus all slots will be generated regardless of whether they have values. After producing values for all domain-slots, the model generates an <EOS> to end the generation process. In practice, we find that the model never omits any of the N_s domain-slot pairs during generation, further confirming GPT-2's ability to produce structured output. An example of input/output is at the bottom of Fig.2 ③: Generation.

Decoding: In generation the model incorporates the GAT features $\mathbf{G}_t[i] \in \mathbf{R}^h$ as shown by the pink arrows in Fig.2 ③: Generation. When predicting the value of the i^{th} domain-slot (in this example the domain-slot is `hotel-name`), the GPT-2 features used for token decoding are concatenated with the domain-slot features $\mathbf{G}_t[i]$ from the output of the GATs. The prediction of the value for each domain-slot will incorporate the domain-slot features produced by the GATs. When predicting tokens that are not related to value predictions (black arrows in the figure), an all-zero tensor is concatenated to keep consistency. The input dimension of the linear layer in decoding is extended to accommodate the GAT features in concatenation.

Fine-tuning: Fine-tuning of GPT-2 for MultiWOZ is done in the usual way. Each turn t in the MultiWOZ training set is transformed into a sequence 'H_t <BOS> Y_t' where Y_t contains the sequence of domain-slots and values for dialogue-turn t, as extracted from the annotated training set. Training proceeds by optimising $P(Y_t|H_t;\theta)$ over the training set.

5 Experiments

We report dialogue state tracking performance on *MultiWOZ 2.0* (Budzianowski et al., 2018) with its multi-domain goal-oriented dialogue conversations and annotations. For direct comparison to the previous literature, we use the same preprocessing as Wu et al. (2019) and Zhou and Small (2019). Two metrics are used for evaluating the model performance:

Slot Accuracy measures the ratio of successful slot value predictions among all the slots of each dialogue turn in ground-truth.

Joint Goal Accuracy compares the predicted belief state to the ground truth at every dialogue turn, and the output is considered correct only if all the predicted slot values exactly match the ground truth values.

5.1 Baseline Performance

We take the performance of several recently published systems as points for comparison: **TRADE** (Wu et al., 2019), **DST-Picklist** (Zhang et al., 2019), and **SUMBT+LaRL** (Lee et al., 2020). These models employ transfer learning, classification with a mixed strategy, and reinforcement learning, respectively. As discussed in Sec. 2, we also compare our model to Graph-based DSTs: **SOM-DST+SG** (Zhu et al., 2020), **GCDST** (Wu et al., 2020), and **SST** (Chen et al., 2020).

In addition, we consider two models as most relevant baselines, and we have attempted to reproduce their results for inclusion here[1]:

DSTQA* (Zhou and Small, 2019)[2]: A bi-LSTM-based DST model utilizing a graph attention network to capture inter-slot relationships, which motivates the architecture introduced in this paper.

SimpleTOD* (Hosseini-Asl et al., 2020)[3]: A GPT-2-based dialogue state tracker, which is similar to our base model, without graph enhancement.

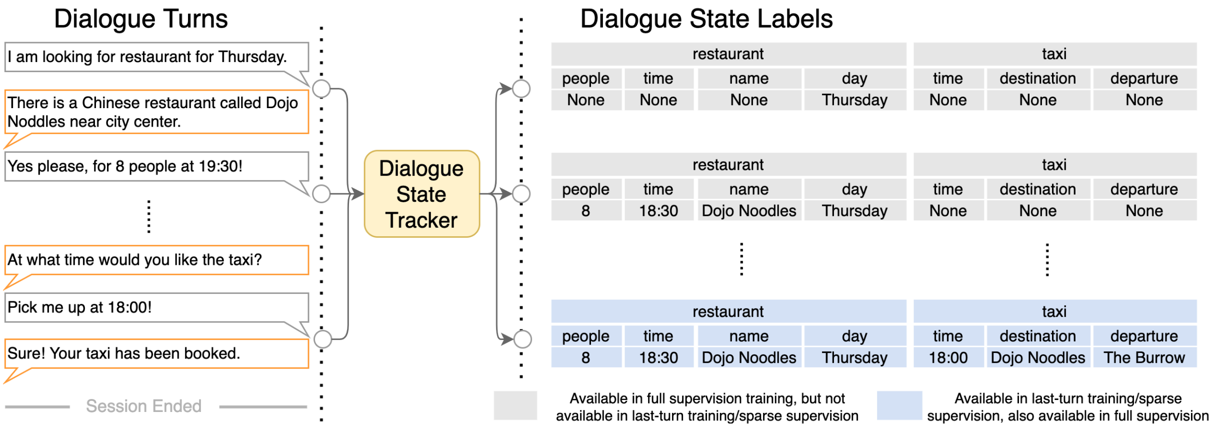
5.2 Training Regimes

We investigate two training scenarios. The first approach is fully supervised at the level of individual turns, following the common practice (e.g. Hosseini-Asl et al.(2020)). The second approach is **Sparsely-Supervised Training**, in which training is at the entire dialogue level, i.e. including only the dialogue state labels at the final turn without their intermediate states during the session, but with the previous dialogue turns included as history (shown in Fig. 3). The two components,

[1] The results shown in this paper might be different from what they reported. See Appendix A.1.

[2] https://github.com/alexa/dstqa

[3] https://github.com/salesforce/simpletod



Dialogue Turns

I am looking for restaurant for Thursday.

There is a Chinese restaurant called Dojo Noddles near city center.

Yes please, for 8 people at 19:30!

⋮

At what time would you like the taxi?

Pick me up at 18:00!

Sure! Your taxi has been booked.

Session Ended

Dialogue State Labels

restaurant				taxi		
people	time	name	day	time	destination	departure
None	None	None	Thursday	None	None	None

restaurant				taxi		
people	time	name	day	time	destination	departure
8	18:30	Dojo Noodles	Thursday	None	None	None

⋮ ⋮

restaurant				taxi		
people	time	name	day	time	destination	departure
8	18:30	Dojo Noodles	Thursday	18:00	Dojo Noodles	The Burrow

▨ Available in full supervision training, but not available in last-turn training/sparse supervision

▨ Available in last-turn training/sparse supervision, also available in full supervision

Figure 3: An example of the sparely-supervised training scenario where only the annotations at the last turn (highlighted in blue) are available.

	Joint (%)	Slot (%)
TRADE	48.62	96.92
DSTQA*	52.24	97.28
SimpleTOD*	51.37	96.48
SOM-DST+SG	52.53	N/A
GCDST	50.68	N/A
SST	51.17	N/A
SUMBT+LaRL	51.52	**97.89**
DST-Picklist	54.39	N/A
L0P0K0-NoGraph	53.00	97.34
L4P4K2-DSGraph	**54.86**	97.47
L4P4K2-DSVGraph	54.62	97.42

(a) Training with all samples.

	Joint (%)	Slot (%)
1. DSTQA*-LastTurn	22.88	93.53
2. SimpleTOD*-LastTurn	48.16	96.31
3. L0P0K0-NoGraph-LastTurn	48.07	96.88
4. L1P1K2-DSGraph-LastTurn	49.00	96.98
5. L1P1K2-DSVGraph-LastTurn	49.25	96.97
6. L1P1K3-DSVGraph-LastTurn	49.93	97.05
7. L4P4K2-DSGraph-LastTurn	**50.43**	**97.14**
8. L4P4K2-DSVGraph-LastTurn	50.26	97.04
9. L4P4K3-DSVGraph-LastTurn	50.05	97.04

(b) Training with only last-turn samples.

Table 1: *MultiWOZ 2.0* Dialogue State Tracking performance comparison, and ablation study. The metrics are joint accuracy (Joint) and slot accuracy (Slot) in %. GAT models are named “L{_}P{_}K{_}-[Graph_Type]”, for number of layers L, number of heads per layer P, and number of hops K (Sec. 4.2).

GPT-2 and **GAT**, are jointly trained. More details are in Appendix A.2. Under sparse supervision, the training set is reduced from $54,971$ turns to only last-turn samples $7,884$ (14.3%); validation utilizes only last-turn samples, as well. Note that evaluation is performed with the standard, MultiWOZ test set ($7,372$ samples) for models trained under either regime. For comparison, we produced the results of **DSTQA*** and **SimpleTOD*** using the same last-turn samples. These are denoted with a “-LastTurn” suffix as in Table 1b.

We denote the configurations of **GAT** with “L{_}P{_}K{_}-[Graph_Type]” format, filling in number of layers L, number of heads per layer P, and number of hops K.

6 DST Performance

We first compare our model with baseline systems. As shown in Table 1a, L0P0K0-NoGraph, which has no graph enhancement, achieves higher joint accuracy than most of the baseline models including the graph-based models such as GCDST and SOM-DST+SG, setting a strong baseline for further improvement to GPT-2-based generation. L4P4K2-$\star$ models, with multiple GAT layers to encourage inter-slot information exchange, show significantly better performance. L4P4K2-DSGraph achieves 54.86% in joint accuracy, highest amongst these systems.

In the sparsely-supervised scenario, the performance of the baseline GPT-2 model drops to 48.07% joint accuracy (L0P0K0-NoGraph, Table 1b). Incorporating GAT in the system (L4P4K2-DSGraph-LastTurn) achieves 50.43% in joint accuracy, leading to a 3% degradation relative to L0P0K0-NoGraph fine-tuned with the full set of annotated dialogue turns. By contrast, DSTQA⋆-LastTurn, which utilizes bidirectional LSTM modules, exhibits a sharp performance decrease to 22.88% joint accuracy; we hypothesize this that the LSTM-based model can not annotate short dialogue samples well having been fine-tuned only with the last-turn samples which have relatively longer dialogue history and annotations.

The sparsely supervised scenario further shows the value of augmenting GPT-2 with representations derived from GATs (Table 1b). Relative to the base system (Model 3, Table 1b), L1P1K2-DSVGraph-LastTurn (Model 5) improves accuracy by incorporating GAT representations in which slot nodes depend on only value nodes. When the number of hops is increased, slot nodes influence each other via intermediate value nodes, yielding further improvement (Models 6,8).

However, multiple GAT layers ($L = 4, P = 4$, Models 7,8,9, Table 1b) do not differ much in performance, showing that dependencies between slots nodes and values can be captured with sufficient layers (thus effectively more hops of information exchange) and attention heads. In particular, although the number of hops (K) is relatively small, feature passing between distant nodes can occur from layer to layer.

We summarize our findings as below:

(1) Through modelling values nodes, the DSVGraph is able to capture dependencies between slots that share values, resulting in a slight improvement over the DSGraph when the number of layers/hops are limited (Table 1b Model 5, 6 v.s. Model 4).

(2) With sufficient layers of GATs, DSGraph compensates for the lack of explicit value nodes and matches and sometimes outperforms the performance of DSVGraph, but this is at the cost of additional modelling complexity (comparing Table 1b Model 8, 9 and Model 7). Understanding these trade-offs will be helpful in applying these models in larger, more complex domains.

In the following sections (Sec. 6.1, 6.2, and 6.3), we investigate how graph modules improve the performance of the base fine-tuned GPT-2 model.

6.1 GATs capture inter-slot dependencies

The accuracy of each domain-slot of several models is shown in Fig. 4. The horizontal axis follows the serialization order of domain-slot pairs in the model output. As discussed in Sec. 1, when predicting <restaurant-area> (position 14), the causal GPT-2 model is able to condition on what has been predicted for <attraction-area> (position 1), but not the other direction, possibly incurring decreased performance for earlier slots. After introducing graph modules this effect of causality is mitigated. For example, as shown in Fig. 4, the slot accuracy of "attraction" domain is always boosted by graph-enhanced models (green and yellow). We further note that these graph-enhanced models perform generally better in those intuitively correlated slots (e.g. <hotel-pricerange> and <restaurant-pricerange>). We conclude that graph-based inter-slot dependencies are beneficial to such GPT-2-based generation models.

6.2 GATs improve the predictions at intermediate dialogue turns

It is important to analyze what impact the last-turn training brings to the predictions at intermediate turns, and how graph modules improve them. A dialogue session might run for 3-4 turns to complete a single task, or up to 18 turns to complete a complex task (e.g. booking a train, taxi, and hotel in the same session). Starting from 0% (the first turn) to 100% (the last turn), we report the prediction accuracy of all slots as the dialogue progresses. As shown in Fig. 5, the baseline model trained with all training samples (L4P4K2-DSGraph) shows a downward trend in prediction accuracy as the dialogue progresses. This agrees with our observation that as dialogue progresses, the domain-slot prediction task becomes larger and more complex (e.g. time-related slots such as taxi-arriveBy are known to be difficult and tend to appear late in a session).

Comparing L4P4K2-DSGraph (blue) and L0P0K0-NoGraph-LastTurn (yellow), the performance throughout the dialogue sessions lags by around 5% in joint accuracy. When graph modules are introduced in models such as L4P4K2-DSGraph-LastTurn and L4P4K2 -DSVGraph-LastTurn, the system performance in the latter half of the dialogue degrades much less. For instance, when towards the end of dialogues (progress higher than 80%), the difference in joint accuracy of L4P4K2-DSGraph (blue) and L4P4K2-DSGraph-LastTurn (brown) is less than 2%. Graph-enhanced models significantly improve the performance in the latter halves of dialogues. A possible reason is that, as

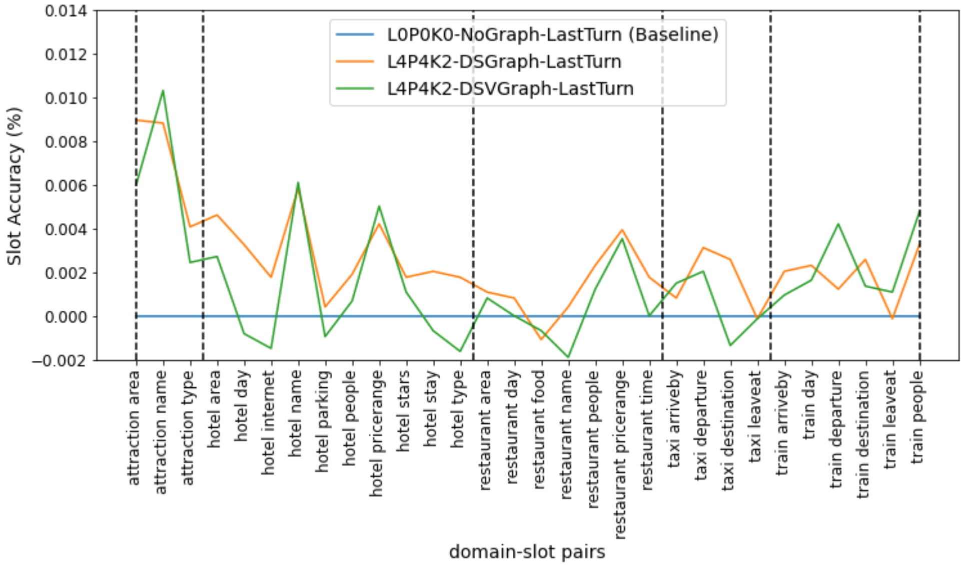


Figure 4: Slot accuracy relative to baseline (`L0P0K0-NoGraph-LastTurn`), in the serialization order for GPT-2 generation. Domain-slot accuracy is improved, particularly for items earlier in the serialisation.

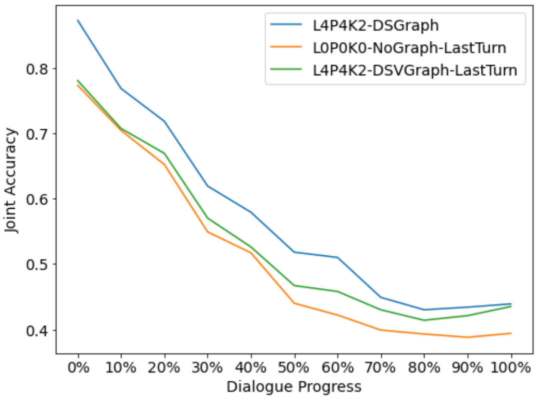


Figure 5: Prediction accuracy against of dialogue progress. Models trained with only last-turn samples can utilize GATs to retain much performance in the latter halves of dialogues (50% to 100%).

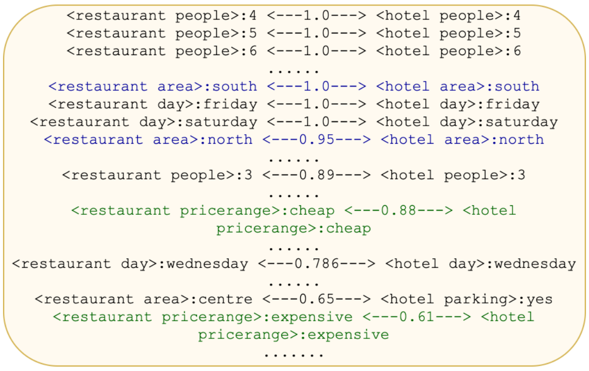

```
      <restaurant people>:4 <---1.0---> <hotel people>:4
      <restaurant people>:5 <---1.0---> <hotel people>:5
      <restaurant people>:6 <---1.0---> <hotel people>:6
                            ......
    <restaurant area>:south <---1.0---> <hotel area>:south
    <restaurant day>:friday <---1.0---> <hotel day>:friday
  <restaurant day>:saturday <---1.0---> <hotel day>:saturday
    <restaurant area>:north <---0.95---> <hotel area>:north
                            ......
      <restaurant people>:3 <---0.89---> <hotel people>:3
                            ......
   <restaurant pricerange>:cheap <---0.88---> <hotel
                   pricerange>:cheap
                            ......
<restaurant day>:wednesday <---0.786---> <hotel day>:wednesday
                            ......
   <restaurant area>:centre <---0.65---> <hotel parking>:yes
 <restaurant pricerange>:expensive <---0.61---> <hotel
                   pricerange>:expensive
                            .......
```

Figure 6: A sample of slot value pairs in the test set with their Jaccard scores. Each entry shows that values in two different slots are bridged by their Jaccord scores. Higher scores indicate stronger dependencies.

the dialogue proceeds, more values are specified and correlated domain-slots appear together more frequently, which enables graph modules to exploit the dependencies between slots.

6.3 GATs improve the predictions of correlated slots

We investigate these inter-slot dependencies and to what extent they affect our graph models.

For every pair of value candidates under two distinct slots (e.g. `<hotel-people>:3` and `<restaurant-people>:3` form a value pair), we measure the correlation of the two values using Jaccard similarity coefficient (Zhang and Srihari, 2003). Jaccard score of two sets C_1 and C_2 is defined as: $J(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$.

For each value pair, we flag their occurrences in the turn-level test set samples where both of their corresponding slots have non-empty annotations. The Jaccard score is then computed from the co-occurrences of the two values. Further details are given in Appendix A.3. Intuitively, the score indicates whether the two values in the pair tend to appear together or not, which is a suitable measurement for value-level dependencies, at the same time bridging the slots to which they belong. Note that these scores are objective values derived from the test set, without the engagement of any model.

Fig. 6 shows value pairs with their Jaccard scores from the test set annotations. There is clear evidence of dependencies in slots across domains. For example, `<restaurant-pricerange>` and

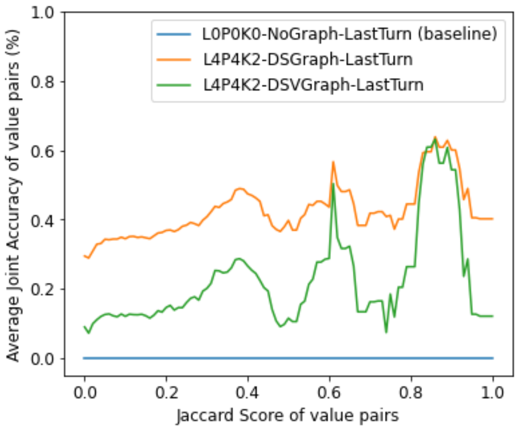


Figure 7: The joint accuracy (relative to baseline `L0P0K0-NoGraph-LastTurn`) changes with the Jaccard Score of value pairs. A moving window of size 0.1 is applied to obtain the averaged joint accuracy around each Jaccard score being evaluated.

`<hotel-pricerange>` are bridged by their values (`cheap` and `expensive`) with high Jacard scores (highlighted in green). The values of `<restaurant-area>` also aligns well with those of `<hotel-area>` (in blue).

We run three models (as shown in the legend of Fig. 7) and for each value pair obtain the average pair accuracy (the success rate of correctly generating both values). We then plot the change in average pair accuracy (relative to baseline values) with the increasing Jaccard coefficient in Fig. 7. Compared to the baseline without GATs (blue), the graph-enhanced models (yellow and green) perform better when predicting values that have high Jaccard scores. Specifically, `L4P4K2-DSVGraph-LastTurn` has a 0.15% boost when Jaccard is around 0.2, and it further improves the performance to 0.6% at the Jaccard value of 0.88. Therefore, we can conclude that the graph modules enable the models to exploit the inter-slot dependencies and learn better in those highly correlated values.

7 Conclusion

We presented a novel hybrid architecture that augments GPT-2 with representations derived from Graph Attention Networks in such a way to allow causal, sequential prediction of slot values. Our analysis shows that these graph-enhanced models mitigate some of the issues that arise in prediction with left-to-right generative models. We also demonstrate that our model can exploit dependencies among domain-slot values, improving accuracy for systems trained with weak supervision.

8 Acknowledgements

We thank Zhilin Wang (University of Washington) for initial discussions and Qingbiao Li (University of Cambridge) for an initial implementation of graph convolution operations. The code of this project is released on Github.[4]

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 7521–7528. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. ACL.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. TripPy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

Jiaying Hu, Yan Yang, Chencai Chen, Liang He, and Zhou Yu. 2020. SAS: Dialogue state tracking via slot attention and slot information sharing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6366–6375, Online. Association for Computational Linguistics.

[4] https://github.com/LinWeizheDragon/Knowledge-Aware-Graph-Enhanced-GPT-2-for-Dialogue-State-Tracking

Yi Huang, Junlan Feng, Min Hu, Xiaoting Wu, Xiaoyu Du, and Shuo Ma. 2020. Meta-reinforced multi-domain state generator for dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7109–7118, Online. ACL.

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

Hwaran Lee, Seokhwan Jo, HyungJun Kim, Sangkeun Jung, and Tae-Yoon Kim. 2020. Sumbt+ larl: End-to-end neural task-oriented dialog system with reinforcement learning. *arXiv preprint arXiv:2009.10447*.

Qingbiao Li, Weizhe Lin, Zhe Liu, and Amanda Prorok. 2021. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540.

Shuailong Liang, Lahari Poddar, and Gyuri Szarvas. 2021. Attention guided dialogue state tracking with sparse supervision. *arXiv preprint arXiv:2101.11958*.

Yawen Ouyang, Moxin Chen, Xinyu Dai, Yinggong Zhao, Shujian Huang, and Jiajun Chen. 2020. Dialogue state tracking with explicit slot connection modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 34–40, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Yong Shan, Zekang Li, Jinchao Zhang, Fandong Meng, Yang Feng, Cheng Niu, and Jie Zhou. 2020. A contextual hierarchical attention network with adaptive objective for dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6322–6333, Online. Association for Computational Linguistics.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. ACL.

Peng Wu, Bowei Zou, Ridong Jiang, and AiTi Aw. 2020. GCDST: A graph-based and copy-augmented multi-domain dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1063–1073, Online. Association for Computational Linguistics.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Bin Zhang and Sargur N Srihari. 2003. Properties of binary vector dissimilarity measures. In *Proc. JCIS Int'l Conf. Computer Vision, Pattern Recognition, and Image Processing*, volume 1.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *9th Joint Conference on Lexical and Computational Semantics (SEM 2020)*.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *In NeurIPS 2019 Workshop on Conversational AI*.

Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020. Efficient context and schema fusion networks for multi-domain dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 766–781, Online. Association for Computational Linguistics.

A Appendices

A.1 Reproduction Details

Dialogue state tracking performance reported in this paper are replications of published results for **DSTQA*** and **SimpleTOD*** using source code accompanying the papers describing these systems. The asterisk indicates results found by our replication.

DSTQA*: We used the software released by Zhou and Small (2019)[5] to retrain and evaluate the system with hyperparameters set as in the original code. Training ran for 300 epochs and 2 days. The best model was found at epoch 174 based on the validation accuracy of all slots.

DSTQA*-LastTurn: We used the same software environment as for **DSTQA***, modified such that only the final turn of training/validation samples was used in training. The training was run for 300 epochs and 20 hours, and the best model was found at epoch 109, after which the model exhibited overfitting and reduced performance.

[5] https://github.com/alexa/dstqa

Sample index	0	1	2	3	4
Labels for `<restaurant-pricerange>`	none	expensive	moderate	expensive	moderate
Labels for `<hotel-pricerange>`	none	moderate	expensive	expensive	cheap
C_1:`<restaurant-pricerange>:expensive`	ignore	1	0	1	0
C_2:`<hotel-pricerange>:expensive`	ignore	0	1	1	0
dependent? $C_1 \cap C_2$	ignore	False	False	True	True

Table 2: Example of calculating Jaccard Scores for the value pair `<restaurant-pricerange>:expensive` and `<hotel-pricerange>:expensive`. Here shows 5 possible turn-level samples to demonstrate how we flag the occurrences for the value pair.

	Joint(%)	Slot(%)
Zhou and Small (2019)	51.44	97.24
DSTQA*	52.24	97.28
DSTQA*-LastTurn	22.88	93.53

Table 3: DSTQA Performance on MultiWOZ 2.0.

	Joint(%)	Slot(%)
SimpleTOD*	51.37	96.48
SimpleTOD*-LastTurn	48.16	96.31

Table 4: Performance of SimpleTOD (Hosseini-Asl et al., 2020) in MultiWOZ 2.0 .

SimpleTOD*: Software was downloaded from the official repository[6] of SimpleTOD. After fixing several bugs according to the discussions in the repository, we evaluated this model in MultiWOZ 2.0 (Budzianowski et al., 2018) for a fair comparison with our proposed models. The best model was found by the perplexity of validation set, as recommended by the paper.

SimpleTOD*-LastTurn: We reduced the training data set to only final turns of dialogues as in **DSTQA*-LastTurn**, and produced the results to compare with our proposed models.

A.2 Training Details

All experiments were done with a RTX3090 GPU. Fine-tuning is done with an AdamW optimizer with a linear decay learning rate for 8 epochs (36 epochs for sparsely-supervised training). Each epoch costs around 1 hour to complete on the GPU used. The GPT-2 component loads the pre-trained parameters of the standard model (12-layer, 768-hidden, 12-heads, 117M parameters, OpenAI GPT-2 English model) provided by huggingface[7]. Though the GPT-2 and $\mathbf{GAT}$ are jointly trained, the initial learning rates are 6.25×10^{-5} and 8×10^{-5} for two major components respectively. Training details can be found in our official Github repository.[8]

A.3 Calculation of Jaccard Scores

Table 2 shows an example of labeling sequences of C_1 and C_2 from which Jaccard scores are computed by $J(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$. With the five samples shown, the Jaccard score is $\frac{2}{4} = 0.5$. The value is not high as intuitively the occurrence and absence of `<restaurant-pricerange>:expensive` does not pair well with those of `<hotel-pricerange>:expensive`. As the number of samples increases, this score effectively reflects how a slot value depends on the other, leading to a good measurement of coreference and dependencies.

[6] https://github.com/salesforce/simpletod

[7] https://huggingface.co/

[8] https://github.com/LinWeizheDragon/Knowledge-Aware-Graph-Enhanced-GPT-2-for-Dialogue-State-Tracking