Slot Self-Attentive Dialogue State Tracking

Fanghua Ye
University College London, UK
fanghua.ye.19@ucl.ac.uk

Jarana Manotumruksa
University College London, UK
j.manotumruksa@ucl.ac.uk

Qiang Zhang
University College London, UK
qiang.zhang.16@ucl.ac.uk

Shenghui Li
Uppsala University, Sweden
shenghui.li@it.uu.se

Emine Yilmaz
University College London, UK
emine.yilmaz@ucl.ac.uk

ABSTRACT

An indispensable component in task-oriented dialogue systems is the dialogue state tracker, which keeps track of users' intentions in the course of conversation. The typical approach towards this goal is to fill in multiple pre-defined slots that are essential to complete the task. Although various dialogue state tracking methods have been proposed in recent years, most of them predict the value of each slot separately and fail to consider the correlations among slots. In this paper, we propose a slot self-attention mechanism that can learn the slot correlations automatically. Specifically, a slot-token attention is first utilized to obtain slot-specific features from the dialogue context. Then a stacked slot self-attention is applied on these features to learn the correlations among slots. We conduct comprehensive experiments on two multi-domain task-oriented dialogue datasets, including MultiWOZ 2.0 and MultiWOZ 2.1. The experimental results demonstrate that our approach achieves state-of-the-art performance on both datasets, verifying the necessity and effectiveness of taking slot correlations into consideration.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Discourse, dialogue and pragmatics**; *Machine learning*; *Supervised learning*; *Learning latent representations.*

KEYWORDS

dialogue state tracking, belief tracking, slot self-attention, task-oriented dialogue system

ACM Reference Format:
Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot Self-Attentive Dialogue State Tracking. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3449939

1 INTRODUCTION

Task-oriented dialogue systems such as Apple Siri and Amazon Alexa work as virtual personal assistants. They can be leveraged to help users complete numerous daily tasks. A typical task-oriented dialogue system consists of four key components, i.e., natural language understanding (NLU), dialogue state tracking (DST), dialogue policy learning (DPL) and natural language generation (NLG) [5, 12]. Among them, DST aims at keeping track of users' intentions at each turn of the dialogue. Since DPL and NLU depend on the results of DST to select the next system action and generate the next system response, an accurate prediction of the dialogue state is crucial to enhance the overall performance of the dialogue system [24, 27]. The typical dialogue state comprises a set of predefined slots and their corresponding values [33] (refer to Table 1 for an example). Therefore, the goal of DST is to predict the values of all slots at each turn based on the dialogue context.

DST has by far attracted much attention from both industry and academia, and numerous DST approaches have been proposed [9, 18, 19, 25, 43, 51]. Although the state-of-the-art DST methods have achieved good performance, most of them predict the value of each slot separately, failing to consider the correlations among slots [6, 22]. This can be problematic, as slots in a practical dialogue are unlikely to be entirely independent. Typically, some slots are highly correlated with each other, demonstrated by coreference and value sharing. Take the dialogue shown in Table 1 as an example. The value of slot "*taxi-arriveby*" is indicated by the slot "*restaurant-book time*". Thus, slot "*taxi-arriveby*" and slot "*restaurant-book time*" share the same value. The value of slot "*taxi-destination*" should

Table 1: An example dialogue with two domains. The value of slot "*taxi-arriveby*" should be inferred according to the value of slot "*restaurant-book time*". The value of slot "*taxi-destination*" is the same as that of slot "*restaurant-name*".

Sys: Hi, what can I do for you?
User: Please find me a Chinese restaurant.
State: *restaurant-food=chinese*
Sys: Charlie Chan fits your criterion, can I book it for you?
User: Yes, I need a table on Monday at 12:15.
State: *restaurant-food=chinese; restaurant-name=charlie chan restaurant-book day=monday; restaurant-book time=12:15*
Sys: Booking is successful. Is there anything else I can assist you with today?
User: I also need a taxi to get me to the restaurant on time.
State: *restaurant-food=chinese; restaurant-name=charlie chan restaurant-book day=monday; restaurant-book time=12:15 taxi-destination=charlie chan; taxi-arriveby=12:15*

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449939

also be taken from slot "*restaurant-name*". Furthermore, slot values can have a high co-occurrence probability. For example, the name of a restaurant should be highly relevant to the food type it serves.

In the literature, we notice that several DST approaches [6, 22, 60] have tried to model the correlations among slots to a certain degree. However, these methods rely on huge human efforts and prior knowledge to determine whether two slots are related or not. As a consequence, they are severely deficient in scalability. Besides, they all leverage only the semantics of slot names to measure the relevance among slots and ignore the co-occurrences of slot values. Utilizing only the slot names is insufficient to capture the slot correlations completely and precisely. On one hand, the correlations among some slots may be overestimated, as slot values in a particular dialogue depend highly on the dialogue context. On the other hand, the correlations among some slots may be underestimated because their names have no apparent connections, even though their values have a high co-occurrence probability.

In this paper, we propose a new DST approach, named **S**lot self-a**T**tentive dialogue st**A**te t**R**acking (**STAR**), which takes both slot names and their corresponding values into account to model the slot correlations more precisely. Specifically, STAR first employs a slot-token attention module to extract slot-specific information for each slot from the dialogue context. It then utilizes a stacked slot self-attention module to learn the correlations among slots in a fully data-driven way. Hence, it does not ask for any human efforts or prior knowledge. The slot self-attention module also provides mutual guidance among slots and enhances the model's ability to deduce appropriate slot values from related slots. We conduct extensive experiments on both MultiWOZ 2.0 [3] and MultiWOZ 2.1 [11] and show that STAR achieves better performance than existing methods that have taken slot correlations into consideration. STAR also outperforms other state-of-the-art DST methods[1].

2 RELATED WORK

DST is crucial to the success of a task-oriented dialogue system. Traditional statistical DST approaches rely on either the semantics extracted by the NLU module [45, 49, 52, 53] or some hand-crafted features and complex domain-specific lexicons [20, 32, 38, 50, 61] to predict the dialogue state. These methods usually suffer from poor scalability and sub-optimal performance. They are also vulnerable to lexical and morphological variations [27, 43].

Owing to the rise of deep learning, a neural DST model called neural belief tracking (NBT) has been proposed [33]. NBT employs convolutional filters over word embeddings in lieu of hand-crafted features to predict slot values. The performance of NBT is much better than previous DST methods. Inspired by this seminal work, a lot of neural DST approaches based on long short-term memory (LSTM) network [34, 40–42, 59] and bidirectional gated recurrent unit (BiGRU) network [22, 31, 35, 39, 55, 57] have been proposed for further improvements. These methods define DST as either a classification problem or a generation problem. Motivated by the advances in reading comprehension [4], DST has been further formulated as a machine reading comprehension problem [13, 14, 30, 31]. Other techniques such as pointer networks [56] and reinforcement learning [7, 8, 23] have also been applied to DST.

[1]Code is available at https://github.com/smartyfh/DST-STAR

Recently, pre-training language models has gained much attention from both industry and academia, and a great variety of pre-trained language models such as BERT [10] and GPT-2 [36] have been released. Since the models are pre-trained on large corpora, they demonstrate strong abilities to produce good results when transferred to downstream tasks. In view of this, the research of DST has been shifted to building new models on top of these powerful pre-trained language models [15, 21, 25, 27, 29, 43, 48, 58]. For example, SUMBT [27] employs BERT to learn the relationships between slots and dialogue utterances through a slot-word attention mechanism. CHAN [43] is built upon SUMBT via taking into account both slot-word attention and slot-turn attention. To better model dialogue behaviors during pre-training, TOD-BERT [54] further pre-trains the original BERT model using several task-oriented dialogue datasets. SOM-DST [25] considers the dialogue state as an explicit fixed-sized memory and selectively overwrites this memory to avoid predicting the dialogue state at each turn from scratch. TripPy [18] uses three copy mechanisms to extract slot values. MinTL [29] exploits T5 [37] and BART [28] as the dialogue utterance encoder and jointly learns dialogue states and system responses. It also introduces Levenshtein belief spans to track dialogue states efficiently. NP-DST [16] and SimpleTOD [21] adopt GPT-2 as the dialogue context encoder and formulate DST as a language generation task.

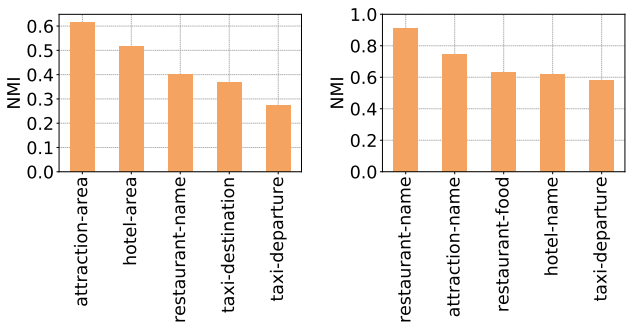
All the methods mentioned above predict the value of each slot separately and ignore the correlations among slots. We notice that several approaches [6, 22, 60] have tried to model the relevance among slots to a certain degree. Specifically, CSFN-DST [60] and SST [6] construct a schema graph to capture the dependencies of different slots. However, the manually constructed schema graph is unlikely to reflect the correlations among slots completely. Besides, lots of prior knowledge is involved during the construction process. Therefore, CSFN-DST and SST are not scalable. SAS [22] calculates a slot similarity matrix to facilitate information flow among similar slots. The similarity matrix is computed based on either the cosine similarity or the K-means clustering results of slot names. However, when computing the similarity matrix, SAS involves several hyperparameters, which are hard to set. SAS also fixes the similarity coefficient at 1 if two slots are considered to be relevant. This is obviously impractical. Except for the model-specific drawbacks of CSFN-DST, SST and SAS, they also share a common limitation: they all measure the slot correlations using only the slot names. This may overlook or overrate the dependencies of some slots. Our method utilizes both slot names and their corresponding values to model slot correlations more precisely.

3 PRELIMINARIES

In this section, we first provide the formal definition of DST and then conduct a simple data analysis to show the high correlations among slots in practical dialogues.

3.1 Problem Statement

The goal of DST is to extract a set of slot value pairs from the system response and user utterance at each turn of the conversation. The combination of these slot value pairs forms a dialogue state, which keeps track of the complete intentions or requirements that have been informed by the user to the system. Formally, let



(a) Restaurant Area (b) Taxi Destination

Figure 1: The top-5 most correlated slots of slot "*restaurant-area*" and slot "*taxi-destination*" analyzed on MultiWOZ 2.1. The slot itself is not counted as a relevant one.

$\mathcal{X} = \{(R_1, U_1), (R_2, U_2), \ldots, (R_T, U_T)\}$ denote a conversation of T turns, where R_t and U_t represent the system response and user utterance at turn t, respectively. Suppose that we have a set of J predefined slots $\mathcal{S} = \{S_1, S_2, \ldots, S_J\}$ with S_j being the j-th slot, then the dialogue state at turn t is defined as $\mathcal{B}_t = \{(S_j, V_j^t) | 1 \le j \le J\}$, where $V_j^t \in \mathcal{V}_j$ denotes the corresponding value of slot S_j. $\mathcal{V}_j$ is the value space of slot S_j. Putting the value spaces of all slots together, we construct an ontology $\mathbb{O} = \{(S_1, \mathcal{V}_1), (S_2, \mathcal{V}_2), \ldots, (S_J, \mathcal{V}_J)\}$. Based on the dialogue context $\mathcal{X}$ and the ontology $\mathbb{O}$, the task of DST is defined as learning a dialogue state tracker $\mathcal{F} : \mathcal{X} \times \mathbb{O} \rightarrow \mathcal{B}$ that can efficiently capture the user's intentions in the course of conversation. According to this definition, we can see that DST is a relatively challenging problem, as it is needed to predict the values of multiple slots at each turn. Besides, the value spaces of some slots may be large, that is, there may be a large number of candidate values for some slots. This phenomenon makes the prediction of dialogue states even more challenging.

It is worth mentioning that in this paper we use the term "slot" to refer to the concatenation of the domain name and the slot name so as to include both domain and slot information. For example, we use "*restaurant-pricerange*" rather than "*pricerange*" to represent the "*pricerange*" slot in the "*restaurant*" domain. This format is useful, especially when a conversation involves multiple domains. It has also been widely adopted by previous works [22, 25, 27, 43].

3.2 Data Analysis

To intuitively verify the strong correlations among slots in practical dialogues, we conduct a simple data analysis on MultiWOZ 2.1 [11], which is a multi-domain task-oriented dialogue dataset. Specifically, we treat every slot pair as two different partitions of the dataset. For each partition, the corresponding slot values are regarded as the cluster labels. Then we calculate the normalized mutual information (NMI) score between the two partitions. Note that we adopt NMI as the measurement of slot correlations, as mutual information can describe more general dependency relationships beyond linear dependence[2]. We illustrate the top-5 most relevant slots of slot "*restaurant-area*" and slot "*taxi-destination*" in Figure 1. Other slots show similar patterns. From Figure 1, we observe that slot "*restaurant-area*" and slot "*taxi-destination*" are indeed highly correlated with some other slots. The relevant slots are not only within the same domain but also across different domains. For example, slot "*taxi-destination*" correlates highly with slot "*restaurant-food*", even though their names have no apparent connections. This observation consolidates our motivation that it is necessary to take into account both slot names and their values.

[2]https://en.wikipedia.org/wiki/Mutual_information

4 STAR: SLOT SELF-ATTENTIVE DST

In this section, we describe our proposed slot self-attentive DST model *STAR* in detail. The overall architecture of STAR is illustrated in Figure 2, which is composed of a BERT-based context encoder module, a slot-token attention module, a stacked slot self-attention module and a slot value matching module.

4.1 Context Encoder

Recently, many pre-trained language models such as BERT [10] and GPT-2 [36] have shown strong abilities to produce good results when transferred to downstream tasks. In view of this, we employ BERT as the context encoder to obtain semantic vector representations of dialogue contexts, slots and values. BERT is a deep bidirectional language representation learning model rooted in Transformer encoders [46]. It can generate token-specific vector representations for each token in the input sentence as well as an aggregated vector representation of the whole sentence. Therefore, we exploit BERT to generate token-specific vector representations for dialogue contexts and aggregated vector representations for both slots and values.

4.1.1 Dialogue Context Encoder. The dialogue utterances at turn t are represented as $D_t = R_t \oplus U_t$, where $\oplus$ is the operation of sequence concatenation. The dialogue history of turn t is denoted as $M_t = D_1 \oplus D_2 \oplus \cdots \oplus D_{t-1}$. Then, the entire dialogue context of turn t is defined as:

$$X_t = [CLS] \oplus M_t \oplus [SEP] \oplus D_t \oplus [SEP], \tag{1}$$

where $[CLS]$ and $[SEP]$ are two special tokens introduced by BERT. The $[CLS]$ token is leveraged to aggregate all token-specific representations and the $[SEP]$ token is utilized to mark the end of a sentence. Since the maximum input length of BERT is restricted to 512 [10], we must truncate X_t if it is too long. The straightforward way is to cut off the early dialogue history and reserve the most recent one in M_t. However, this operation may throw away some key information. To reduce information loss, we use the previous dialogue state as input as well, which is expected to keep all the slot-related history information. The dialogue state of previous turn is represented by $B_{t-1} = \bigoplus_{(S_j, V_j^{t-1}) \in \mathcal{B}_{t-1}, V_j^{t-1} \neq \text{none}} S_j \oplus V_j^{t-1}$. Note that in B_{t-1} we only include the slots that have been mentioned before (i.e., only non-none slots are considered). By treating B_{t-1} as part of the dialogue history, the entire dialogue context of turn t is finally denoted as[3]:

$$X_t = [CLS] \oplus M_t \oplus B_{t-1} \oplus [SEP] \oplus D_t \oplus [SEP]. \tag{2}$$

[3]Although the previous dialogue state B_{t-1} can serve as an explicit, compact and informative representation of the dialogue history, we find that it is still useful to take the dialogue history M_t as part of the input.

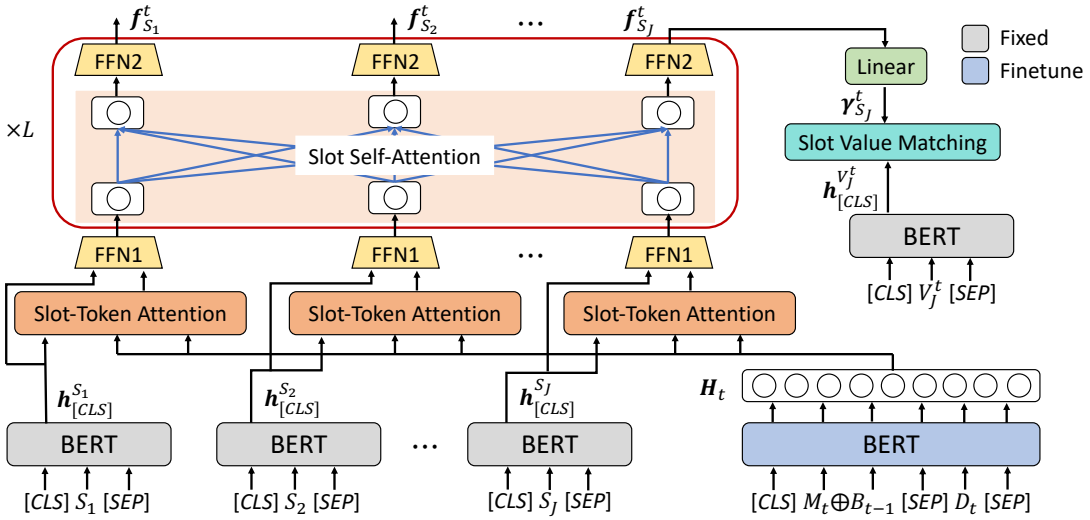


Figure 2: The architecture of our approach STAR. A fine-tuning BERT is used to encode dialogue contexts, another fixed BERT is utilized to generate aggregated vector representations for slots and values. For simplicity, layer normalization and residual connection are omitted, and only the value matching of slot S_J is included. Both FFN1 and FFN2 are feed-forward networks.

Let $|X_t|$ be the number of tokens in X_t. Our first goal is to generate a contextual d-dimensional vector representation for each token in X_t. Let $\boldsymbol{h}_j^t \in \mathbb{R}^d$ denote the vector representation of the j-th token and $\boldsymbol{H}_t = [\boldsymbol{h}_1^t, \boldsymbol{h}_2^t, \ldots, \boldsymbol{h}_{|X_t|}^t] \in \mathbb{R}^{d \times |X_t|}$ the matrix form of all tokens' representations. We simply feed X_t to BERT to obtain $\boldsymbol{H}_t$. Hence, we have:

$$\boldsymbol{H}_t = BERT_{finetune}(X_t). \tag{3}$$

Note that BERT in Eq. (3) will be fine-tuned during training.

4.1.2 Slot and Value Encoder. Following previous works [27, 43], we use another BERT to encode slots and their candidate values. Unlike dialogue contexts, we need to generate aggregated vector representations for slots and values. To achieve this goal, we use the vector representation corresponding to the special token $[CLS]$ to represent the aggregated representation of the whole input sequence. As thus, for any slot $S_j \in \mathcal{S} (1 \le j \le J)$ and any value $V_j^t \in \mathcal{V}_j$, we have:

$$\begin{aligned} \boldsymbol{h}_{[CLS]}^{S_j} &= BERT_{fixed}([CLS] \oplus S_j \oplus [SEP]), \\ \boldsymbol{h}_{[CLS]}^{V_j^t} &= BERT_{fixed}([CLS] \oplus V_j^t \oplus [SEP]), \end{aligned} \tag{4}$$

where $BERT_{fixed}$ means that the pre-trained BERT without fine-tuning is adopted. Fixing the weights of BERT when encoding slots and values is beneficial. Firstly, the slot and value representations can be computed off-line, which reduces the model size of our approach. Secondly, since our model relies on the value representations to score each candidate value of a given slot, fixing the representations of values can reduce the difficulty of choosing the best candidate value.

4.2 Slot-Token Attention

Since there are multiple slots to be predicted at each turn t from the same dialogue context X_t, it is necessary to extract slot-specific information for each slot $S_j (1 \le j \le J)$. Our model employs a multi-head attention mechanism [46] to retrieve the relevant information corresponding to each slot S_j.

4.2.1 Multi-Head Attention. For self-consistency, we provide a brief description of the multi-head attention mechanism. Assume that we are provided with a query matrix $\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{|Q|}] \in \mathbb{R}^{d_{model} \times |Q|}$, a key matrix $\boldsymbol{K} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_{|K|}] \in \mathbb{R}^{d_{model} \times |K|}$ and a value matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{|K|}] \in \mathbb{R}^{d'_{model} \times |K|}$. There are $|Q|$ query vectors, $|K|$ key vectors and $|K|$ value vectors, respectively. Note that the query vectors and the key vectors share the same dimensionality. For each query vector $\boldsymbol{q}_i (1 \le i \le |Q|)$, the attention vector $\boldsymbol{a}_i$ over $\boldsymbol{K}$ and $\boldsymbol{Z}$ with N heads is calculated as follows:

$$e_{ij}^n = \frac{(\boldsymbol{W}_Q^n \boldsymbol{q}_i)^T (\boldsymbol{W}_K^n \boldsymbol{k}_j)}{\sqrt{d_{model}/N}}, \quad \tau_{ij}^n = \frac{exp(e_{ij}^n)}{\sum_{l=1}^{|K|} exp(e_{il}^n)},$$

$$\boldsymbol{a}_i^n = \sum_{j=1}^{|K|} \tau_{ij}^n (\boldsymbol{W}_Z^n \boldsymbol{z}_j), \quad \boldsymbol{a}_i = \boldsymbol{W}_O Concat(\boldsymbol{a}_i^1, \boldsymbol{a}_i^2, \ldots, \boldsymbol{a}_i^N),$$

where $1 \le n \le N$, $\boldsymbol{a}_i \in \mathbb{R}^{d'_{model}}$, $\boldsymbol{W}_Q^n \in \mathbb{R}^{(d_{model}/N) \times d_{model}}$, $\boldsymbol{W}_K^n \in \mathbb{R}^{(d_{model}/N) \times d_{model}}$, $\boldsymbol{W}_Z^n \in \mathbb{R}^{(d'_{model}/N) \times d'_{model}}$ and $\boldsymbol{W}_O \in \mathbb{R}^{d'_{model} \times d'_{model}}$. Putting all the attention vectors together, we obtain the attention matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{|Q|}] \in \mathbb{R}^{d'_{model} \times |Q|}$. The entire process is formulated as:

$$\boldsymbol{A} = MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{Z}).$$

4.2.2 Multi-Head Slot-Token Attention. Our model adopts the multi-head attention mechanism to calculate a d-dimensional vector for each slot S_j as the slot-specific information. More concretely, the slot representation $\boldsymbol{h}_{[CLS]}^{S_j}$ is treated as the query vector, and the dialogue context representation $\boldsymbol{H}_t$ is regarded as both the key matrix and the value matrix. Consequently, the token-level relevance

between slot S_j and dialogue context X_t is summarized as:

$$\boldsymbol{r}_{S_j}^t = MultiHead(\boldsymbol{h}_{[CLS]}^{S_j}, \boldsymbol{H}_t, \boldsymbol{H}_t), \tag{5}$$

where $\boldsymbol{r}_{S_j}^t \in \mathbb{R}^d$. Considering that $\boldsymbol{r}_{S_j}^t$ only contains the value information of slot S_j, we concatenate $\boldsymbol{r}_{S_j}^t$ and $\boldsymbol{h}_{[CLS]}^{S_j}$ to retain its name information. This merged vector is further transformed by a feed-forward neural network as below:

$$\boldsymbol{c}_{S_j}^t = \boldsymbol{W}_2^r ReLU(\boldsymbol{W}_1^r Concat(\boldsymbol{h}_{[CLS]}^{S_j}, \boldsymbol{r}_{S_j}^t) + \boldsymbol{b}_1^r) + \boldsymbol{b}_2^r, \tag{6}$$

where $\boldsymbol{W}_1^r \in \mathbb{R}^{d\times 2d}$, $\boldsymbol{W}_2^r \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_1^r, \boldsymbol{b}_2^r, \boldsymbol{c}_{S_j}^t \in \mathbb{R}^d$.

4.3 Slot Self-Attention

Albeit the slot-token attention is expected to retrieve slot-specific information for all slots, it may fail to capture the valid information of some slots due to the various expressing forms in natural conversations (e.g., coreference, synonymity and rephrasing). In addition, the slot-specific vector $\boldsymbol{c}_{S_j}^t$ of each slot S_j is computed separately. The correlations among slots are ignored. As a result, once the vector $\boldsymbol{c}_{S_j}^t$ doesn't capture the relevant information of slot S_j properly, the model has no chance to deduce the right value for slot S_j. To alleviate this problem, we propose exploiting the slot self-attention mechanism to rectify each slot-specific vector based on the vectors corresponding to all slots. This mechanism should be rational because of the high correlations among slots. Therefore, our model is expected to provide mutual guidance among slots and learn the slot correlations automatically.

The slot self-attention is also a multi-head attention. Specifically, this module is composed of L identical layers and each layer has two sub-layers. The first sub-layer is the slot self-attention layer. The second sub-layer is a feed-forward network (FFN) with two fully connected layers and a ReLU activation in between. Each sub-layer precedes its main functionality with layer normalization [1] and follows it with a residual connection [17].

Let $\boldsymbol{C}_t = [\boldsymbol{c}_{S_1}^t, \boldsymbol{c}_{S_2}^t, \ldots, \boldsymbol{c}_{S_J}^t] \in \mathbb{R}^{d\times J}$ denote the matrix representation of all slot-specific vectors and let $\boldsymbol{F}_t^1 = \boldsymbol{C}_t$. Then, for the l-th slot self-attention sub-layer ($1 \le l \le L$), we have:

$$\begin{aligned} \tilde{\boldsymbol{F}}_t^l &= LayerNorm(\boldsymbol{F}_t^l), \\ \boldsymbol{G}_t^l &= MultiHead(\tilde{\boldsymbol{F}}_t^l, \tilde{\boldsymbol{F}}_t^l, \tilde{\boldsymbol{F}}_t^l) + \tilde{\boldsymbol{F}}_t^l. \end{aligned} \tag{7}$$

In the slot self-attention sub-layer, $\tilde{\boldsymbol{F}}_t^l$ serves as the key matrix, the value matrix, and also the query matrix. For the l-th feed-forward sub-layer, we have:

$$\begin{aligned} \tilde{\boldsymbol{G}}_t^l &= LayerNorm(\boldsymbol{G}_t^l), \\ \boldsymbol{F}_t^{l+1} &= FFN(\tilde{\boldsymbol{G}}_t^l) + \tilde{\boldsymbol{G}}_t^l, \end{aligned} \tag{8}$$

where the function $FFN(\cdot)$ is parameterized by $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^d$, i.e., $FFN(\boldsymbol{y}) = \boldsymbol{W}_2 ReLU(\boldsymbol{W}_1\boldsymbol{y} + \boldsymbol{b}_1) + \boldsymbol{b}_2$.

The final slot-specific vectors are contained in the output of the last layer, i.e., $\boldsymbol{F}_t^{L+1}$. Let $\boldsymbol{F}_t^{L+1} = [\boldsymbol{f}_{S_1}^t, \boldsymbol{f}_{S_2}^t, \ldots, \boldsymbol{f}_{S_J}^t]$, where $\boldsymbol{f}_{S_j}^t \in \mathbb{R}^d$ is the j-th column of $\boldsymbol{F}_t^{L+1}$. $\boldsymbol{f}_{S_j}^t$ is taken as the final slot-specific vector of slot S_j, which is expected to be close to the semantic vector representation of the groundtruth value of slot S_j at turn t. Since the output of BERT is normalized by layer normalization, we also feed $\boldsymbol{f}_{S_j}^t$ to a layer normalization layer, which is preceded by a linear transformation layer as follows:

$$\boldsymbol{\gamma}_{S_j}^t = LayerNorm(Linear(\boldsymbol{f}_{S_j}^t)), \tag{9}$$

where $\boldsymbol{\gamma}_{S_j}^t \in \mathbb{R}^d$.

4.4 Slot Value Matching

To predict the value of each slot $S_j (1 \le j \le J)$, we compute the distance between $\boldsymbol{\gamma}_{S_j}^t$ and the semantic vector representation of each value $V_j' \in \mathcal{V}_j$, where $\mathcal{V}_j$ denotes the value space of slot S_j. Then the value with the smallest distance is chosen as the prediction of slot S_j. We adopt ℓ_2 norm as the distance metric.

During the training phase, we calculate the probability of the groundtruth value V_j^t of slot S_j at turn t as:

$$p(V_j^t|X_t, S_j) = \frac{exp\left(-\|\boldsymbol{\gamma}_{S_j}^t - \boldsymbol{h}_{[CLS]}^{V_j^t}\|_2\right)}{\sum_{V_j'\in\mathcal{V}_j} exp\left(-\|\boldsymbol{\gamma}_{S_j}^t - \boldsymbol{h}_{[CLS]}^{V_j'}\|_2\right)}. \tag{10}$$

Our model is trained to maximize the joint probability of all slots, i.e., $\Pi_{j=1}^J p(V_j^t|X_t, S_j)$. For this purpose, the loss function at each turn t is defined as the sum of the negative log-likelihood:

$$\mathcal{L}_t = \sum_{j=1}^{J} -\log\left(p(V_j^t|X_t, S_j)\right). \tag{11}$$

5 EXPERIMENTAL SETUP

5.1 Datasets

Table 2: Data statistics of MultiWOZ 2.0 & 2.1. The bottom row summarizes the number of dialogues in each domain.

Domain	Attraction	Hotel	Restaurant	Taxi	Train
Slot	area name type	area book day book people book stay internet name parking pricerange stars type	area book day book people book time food name pricerange	arriveby departure destination leaveat	arriveby book people day departure destination leaveat
Train	2717	3381	3813	1654	3103
Validation	401	416	438	207	484
test	395	394	437	195	494

We evaluate our approach on MultiWOZ 2.0 [3] and MultiWOZ 2.1 [11], which are two of the largest publicly available task-oriented dialogue datasets. MultiWOZ 2.0 consists of 10,348 multi-turn dialogues, spanning over 7 domains {*attraction*, *hotel*, *restaurant*, *taxi*, *train*, *hospital*, *police*}. Each domain has multiple predefined slots and there are 35 domain slot pairs in total. MultiWOZ 2.1 is a refined version of MultiWOZ 2.0. According to [11], about 32% of the state annotations have been corrected in MultiWOZ 2.1. Since *hospital* and *police* are not included in the validation set and test set, following previous works [25, 27, 43, 55, 60], we use only the remaining 5 domains in the experiments. The resulting datasets contain 17

distinct slots and 30 domain slot pairs. The detailed statistics are summarized in Table 2.

We follow similar data preprocessing procedures as [55] to preprocess both MultiWOZ 2.0 and MultiWOZ 2.1. And we create the ontology by incorporating all the slot values that appear in the datasets. We notice that several works [27, 43] exploit the original ontology provided by MultiWOZ 2.0 and MultiWOZ 2.1 to preprocess the datasets in their experiments. However, the original ontology is incomplete. If a slot value is out of the ontology, this value is ignored directly in [27, 43], which is impractical and leads to unreasonably high performance.

5.2 Comparison Methods

We compare our model STAR with the following state-of-the-art DST approaches:

- **SST:** SST [6] first constructs a schema graph and then utilizes a graph attention network (GAT) [47] to fuse information from dialogue utterances and the schema graph.
- **SAS:** SAS [22] calculates a binary slot similarity matrix to control information flow among similar slots. The similarity matrix is computed via either a fixed combination method or a K-means sharing method.
- **CREDIT-RL:** CREDIT-RL [7] employs a structured representation to represent dialogue states and casts DST as a sequence generation problem. It also uses a reinforcement loss to fine-tune the model.
- **STARC:** STARC [13] reformulates DST as a machine reading comprehension problem and adopts several reading comprehension datasets as auxiliary information to train the model.
- **CSFN-DST:** Similar to SST, CSFN-DST [60] also constructs a schema graph to model the dependencies among slots. However, CSFN-DST utilizes BERT [10] rather than GAT to encode dialogue utterances.
- **SOM-DST:** SOM-DST [25] regards the dialogue state as an explicit fixed-sized memory and proposes selectively overwriting this memory at each turn.
- **CHAN:** CHAN [43] proposes a slot-turn attention mechanism to make full use of the dialogue history. It also designs an adaptive objective to alleviate the data imbalance issue.
- **TripPy:** TripPy [18] leverages three copy mechanisms to extract slot values from user utterances, system inform memory and previous dialogue states.
- **NP-DST:** NP-DST [16] transforms DST into a language generation problem and adopts GPT-2 [36] as both the dialogue context encoder and the sequence generator.
- **SimpleTOD:** SimpleTOD [21] is also based on GPT-2. Its model architecture is similar to NP-DST.
- **MinTL:** MinTL [29] is an effective transfer learning framework for task-oriented dialogue systems. It introduces Levenshtein belief span to track dialogue states. MinTL uses both T5 [37] and BART [28] as pre-trained backbones. We name them **MinTL-T5** and **MinTL-BART** for distinction.

5.3 Evaluation Metric

We adopt joint goal accuracy [34] as the evaluation metric. Joint goal accuracy is defined as the ratio of dialogue turns for which the value of each slot is correctly predicted. If a slot has not been mentioned yet, its groundtruth value is set to none. All the none slots also need to be predicted. Joint goal accuracy is a relatively strict evaluation metric. Even though only one slot at a turn is mispredicted, the joint goal accuracy of this turn is 0. Thus, the joint goal accuracy of a turn takes the value either 1 or 0.

5.4 Training Details

We employ the pre-trained BERT-base-uncased model[4] as the dialogue context encoder. This model has 12 layers with 768 hidden units and 12 self-attention heads. We also utilize another BERT-base-uncased model as the slot and value encoder. For the slot and value encoder, the weights of the pre-trained BERT model are frozen during training. For the slot-token attention and slot self-attention, we set the number of attention heads to 4. The number of slot self-attention layers (i.e., L) is fixed at 6. We treat the context encoder part of our model as an encoder and the remaining part as a decoder. The hidden size of the decoder (i.e., d) is set to 768, which is the same as the dimensionality of BERT outputs. The BertAdam [26] is adopted as the optimizer and the warmup proportion is fixed at 0.1. Considering that the encoder is a pre-trained BERT model while the decoder needs to be trained from scratch, we use different learning rates for the two parts. Specifically, the peak learning rate is set to 1e-4 for the decoder and 4e-5 for the encoder. We use a training batch size of 16 and set the dropout [44] probability to 0.1. We also exploit the word dropout technique [2] to partially mask the dialogue utterances by replacing some tokens with a special token $[UNK]$. The word dropout rate is set to 0.1. Note that we do not use word dropout on the previous dialogue state, even though it is part of the input. The maximum input sequence length is set to 512. The best model is chosen according to the performance on the validation set. We run the model with different random seeds and report the average results. For MultiWOZ 2.0 and MultiWOZ 2.1, we apply the same hyperparameter settings.

6 EXPERIMENTAL RESULTS

6.1 Baseline Comparison

The joint goal accuracy of our model and various baselines on the test sets of MultiWOZ 2.0 and MultiWOZ 2.1 are shown in Table 3[5], in which we also summarize several key differences of these models. As can be seen, our approach consistently outperforms all baselines on both MultiWOZ 2.0 and MultiWOZ 2.1. Compared with the three methods that have taken slot correlations into consideration (i.e., SST, SAS and CSFN-DST), our approach achieves 2.96% and 1.13% absolute performance promotion on MultiWOZ 2.0 and MultiWOZ 2.1, respectively. Our approach also outperforms other baselines by 1.47% and 1.07% separately on the two datasets. From Table 3, we observe that SST and TripPy are the best performing baselines. Both methods reach higher than 55% joint goal accuracy on MultiWOZ 2.1. However, SST needs to construct a schema graph by involving some prior knowledge manually. The schema graph is exploited to

[4] https://huggingface.co/transformers/model_doc/bert.html

[5] We noted that both CHAN and SimpleTOD ignore the "*dontcare*" slots in the released source codes, which leads to unreasonably high performance. For a fair comparison, we reproduced the results with all "*dontcare*" slots being considered. Our approach can achieve 60% joint goal accuracy if ignoring the "*dontcare*" slots as well.

Table 3: Joint goal accuracy of various methods on the test sets of MultiWOZ 2.0 and MultiWOZ 2.1 (± denotes the standard deviation). † indicates the results reported in the original papers. ⋆ means the results reproduced by us using the source codes. ‡ demonstrates a statistically significant improvement to the best baseline at the 0.01 level using a paired two-sided t-test.

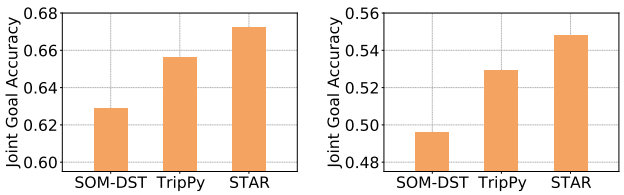
Model	Context Encoder	Dialogue History	Extra Information	Slot Correlations	Joint Goal Accuracy (%)	
					MultiWOZ 2.0	MultiWOZ 2.1
SST†	GAT	Previous Turn	Schema Graph	✓	51.17	55.23
SAS†	BiGRU	Previous μ Turns	-	✓	51.03	-
CREDIT-RL†	BiGRU	Previous μ Turns	-	✗	51.68	50.61
STARC†	BERT	Full History	Auxiliary Datasets	✗	-	49.48
CSFN-DST†	BERT	No History	Schema Graph	✓	51.57	52.88
SOM-DST†	BERT	Previous Turn	-	✗	51.72	53.01
CHAN⋆	BERT	Full History	-	✗	53.06	53.38
TripPy†	BERT	Full History	Action+Label Map	✗	-	55.29
NP-DST †	GPT-2	Full History	-	✗	44.03	-
SimpleTOD⋆	GPT-2	Full History	-	✗	-	51.89
MinTL-T5†	T5	Previous μ Turns	-	✗	52.07	52.52
MinTL-BART†	BART-Large	Previous μ Turns	-	✗	52.10	53.62
STAR	BERT	Full History	-	✓	**54.53±0.21**‡	**56.36±0.34**‡

Table 4: Performance comparison between TripPy and STAR with and without the label map being used on MultiWOZ 2.1.

Model	Label Map	Joint Goal Accuracy (%)
TripPy−	✗	44.90
TripPy	✓	55.29
STAR	✗	56.36
STAR+	✓	**56.58**

capture the correlations among slots. Even though SST leverages some prior knowledge, it is still inferior to our approach. This is because the schema graph only considers the relationships among slot names and thus cannot describe the slot correlations completely. It is worth mentioning that SST also shows a deficiency in utilizing dialogue history. SST achieves the best performance when only the previous turn dialogue history is considered [6]. TripPy shows the best performance among BERT-based baselines. However, it employs both system actions and a label map as extra supervision. The label map is created according to the labels in the training portion of the dataset. During the testing phase, the label map is leveraged to correct the predictions (e.g., mapping "*downtown*" to "*centre*"). The label map is useful, but it may oversmooth some predictions. On the contrary, our model doesn't rely on any extra information and is a fully data-driven approach. Hence, our model is more general and more scalable. Since our model also achieves better performance than SST and TripPy, we can conclude that it is beneficial to take the slot correlations into consideration. The slot self-attention mechanism proposed by us is able to capture the relevance among slots in a better way.

We conduct a further comparison between TripPy and our model STAR with and without the label map being leveraged on MultiWOZ 2.1. We denote TripPy as **TripPy**− when the label map is removed and represent STAR as **STAR**+ when the label map is involved. The results are reported in Table 4. As can be observed,



(a) Single Domain (b) Multi Domain

Figure 3: Joint goal accuracy of single-domain dialogues and multi-domain dialogues on the test set of MultiWOZ 2.1.

the performance of TripPy degrades dramatically if the label map is ignored. However, the label map doesn't have significant impacts on the performance of our approach. With the label map being considered, our approach only shows slightly better performance.

6.2 Single-Domain and Multi-Domain Joint Goal Accuracy

Considering that a practical dialogue may involve multiple domains or just a single domain, it is useful to explore how our approach performs in each scenario. To this end, we report the joint goal accuracy of single-domain dialogues and multi-domain dialogues on the test set of MultiWOZ 2.1, respectively. The results are shown in Figure 3, from which we observe that our approach achieves better performance in both scenarios. The results indicate that our approach can capture the correlations among slots both within a single domain and across different domains. From Figure 3, we also observe that all the methods demonstrate higher performance in the single-domain scenario. Our approach even reaches about 67% joint goal accuracy. While the performance of all methods in the multi-domain scenario slightly goes down, compared to the overall joint goal accuracy shown in Table 3. Nonetheless, our approach still achieves about 55% joint goal accuracy. The results further confirm the effectiveness of our approach.

Table 5: Domain-specific accuracy (%) on MultiWOZ 2.1.

Domain	CSFN-DST	SOM-DST	TripPy	**STAR**
Attraction	64.78	69.83	**73.37**	70.95
Hotel	46.29	49.53	50.21	**52.99**
Restaurant	64.64	65.72	**70.47**	69.17
Taxi	47.35	59.96	37.54	**66.67**
Train	69.79	70.36	72.51	**75.10**

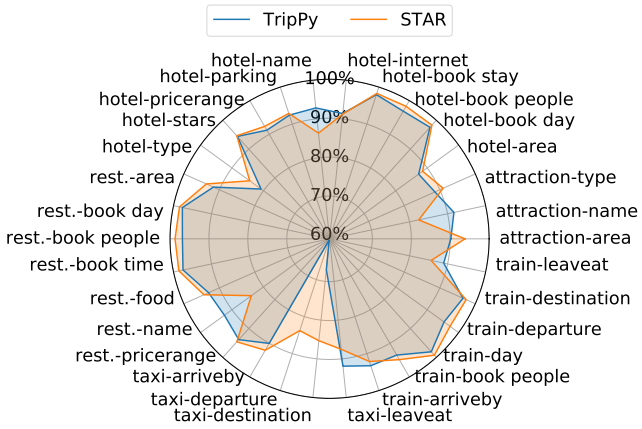


Figure 4: Slot-specific accuracy on MultiWOZ 2.1. The domain "*restaurant*" is represented as "*rest.*" for short.

6.3 Domain-Specific Joint Goal Accuracy and Slot-Specific Accuracy

In this part, we first investigate the performance of our model in each domain. The domain-specific joint goal accuracy on MultiWOZ 2.1 is reported in Table 5, where we compare our approach with CSFN-DST, SOM-DST and TripPy. The domain-specific accuracy is calculated on a subset of the predicted dialogue state. The subset consists of all the slots specific to a domain. In addition, only the domain-active dialogues are considered for each domain. As shown in Table 5, our approach consistently outperforms CSFN-DST and SOM-DST in all domains. Our approach also outperforms TripPy in three domains. Although TripPy demonstrates better performance in the "*attraction*" domain and "*restaurant*" domain, it shows the worst performance in the "*taxi*" domain. As analyzed in [25], the "*taxi*" domain is the most challenging one. This domain also has the least number of training dialogues (refer to Table 2). Owing to the strong capability of modeling slot correlations, our approach achieves much better performance in this challenging domain.

We then illustrate the slot-specific accuracy of our approach and TripPy in Figure 4. The corresponding exact numbers are shown in Table 6 in Appendix A. The slot-specific accuracy measures the accuracy of each individual slot. Note that the slot-specific accuracy is calculated using only the dialogues that involve the domain the slot belongs to. From Figure 4, we observe that both methods demonstrate high performance for most slots. However, TripPy shows relatively poor performance for slot "*taxi-departure*" and slot "*taxi-destination*". The results are consistent with the domain-specific accuracy and explain why TripPy fails in the "*taxi*" domain.

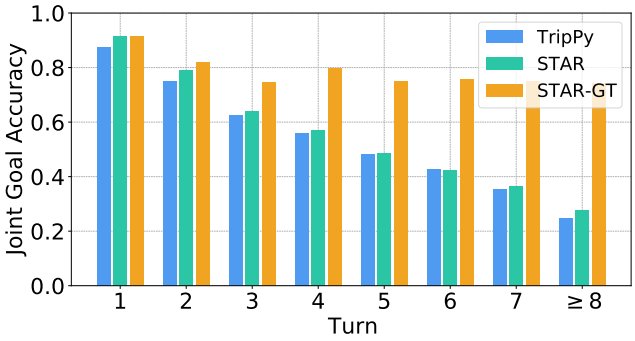


Figure 5: Per-turn joint goal accuracy on MultiWOZ 2.1.

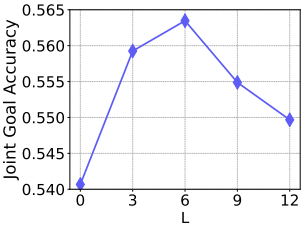


Figure 6: Effects of number of slot self-attention layers.

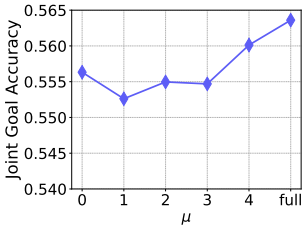


Figure 7: Effects of number of previous dialogue turns.

From Figure 4, we also observe that our approach is inferior to TripPy in the *name*-related slots (i.e., "*attraction-name*", "*hotel-name*" and "*restaurant-name*") and *leaveat*-related slots (i.e., "*taxi-leaveat*" and "*train-leaveat*"). The values of these slots are usually informed by the users explicitly. Since TripPy leverages copy mechanisms to extract values directly, it seems to be more appropriate for these slots. This observation inspires us that it should be beneficial to extend our model by incorporating the copy mechanism, which we leave as our future work.

6.4 Per-Turn Joint Goal Accuracy

Given that practical dialogues have a different number of turns and longer dialogues tend to be more challenging, we further analyze the relationship between the depth of conversation and accuracy of our model. The per-turn accuracy on MultiWOZ 2.1 is shown in Figure 5. For comparison, we also include the results of TripPy and **STAR-GT**. STAR-GT means the groundtruth previous dialogue state is used as the input at each turn. Figure 5 shows that the accuracy of both TripPy and STAR decreases with the increasing of dialogue turns. In contrast, the performance of STAR-GT is relatively stable. This is because errors occurred in early turns will be accumulated to later turns in practice. However, when the groundtruth previous dialogue state is used, there is no error accumulation.

6.5 Effects of Number of Slot Self-Attention Layers

We vary the number of slot self-attention layers (i.e., L) in the range of $\{0, 3, 6, 9, 12\}$ to study its impacts on the performance of our model. The results on MultiWOZ 2.1 are illustrated in Figure 6, from which we observe that our model achieves the best performance when L is set to 6. The performance degrades while L goes larger.

This may be caused by overfitting. Figure 6 also shows that when there is no slot self-attention (i.e., $L = 0$), the joint goal accuracy decreases to around 54%. Note that when $L = 0$, our model doesn't learn the slot correlations any more. Hence, we conclude that it is essential to take the dependencies among slots into consideration.

6.6 Effects of Number of Previous Dialogue Turns

To evaluate the effects of the number of previous dialogue turns (i.e., μ), we vary μ in the range of $\{0, 1, 2, 3, 4, full\}$, where $full$ means all the previous dialogue turns are considered. The results on MultiWOZ 2.1 are shown in Figure 7. As can be seen, when full dialogue history is leveraged, our model demonstrates the best performance. When no dialogue history is employed, our model also reaches higher than 55.5% joint goal accuracy. However, when μ is set to 1, 2 and 3, the performance degrades slightly. This is probably because the incomplete history leads to confusing information and makes it more challenging to extract the appropriate slot values.

7 CONCLUSION

In this paper, we have presented a novel DST approach STAR to modeling the correlations among slots. STAR first employs a slot-token attention to retrieve slot-specific information for each slot from the dialogue context. It then leverages a stacked slot self-attention to learn the dependencies among slots. STAR is a fully data-driven approach. It does not ask for any human efforts or prior knowledge when measuring the slot correlations. In addition, STAR considers both slot names and their corresponding values to model the slot correlations more precisely. To evaluate the performance of STAR, we have conducted a comprehensive set of experiments on two large multi-domain task-oriented dialogue datasets MultiWOZ 2.0 and MultiWOZ 2.1. The results show that STAR achieves state-of-the-art performance on both datasets. For future work, we intend to incorporate the copy mechanism into STAR to enhance its performance further.

ACKNOWLEDGMENTS

This project was funded by the EPSRC Fellowship titled "Task Based Information Retrieval" and grant reference number EP/P024289/1.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[2] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015).
[3] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 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*. 5016–5026.
[4] Danqi Chen. 2018. *Neural reading comprehension and beyond*. Stanford University.
[5] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter* 19, 2 (2017), 25–35.
[6] Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-Guided Multi-Domain Dialogue State Tracking with Graph Attention Neural Networks.. In *AAAI*. 7521–7528.
[7] Zhi Chen, Lu Chen, Zihan Xu, Yanbin Zhao, Su Zhu, and Kai Yu. 2020. CREDIT: Coarse-to-Fine Sequence Generation for Dialogue State Tracking. *arXiv preprint arXiv:2009.10435* (2020).
[8] Zhi Chen, Lu Chen, Xiang Zhou, and Kai Yu. 2020. Deep Reinforcement Learning for On-line Dialogue State Tracking. *arXiv preprint arXiv:2009.10321* (2020).
[9] Yinpei Dai, Huihua Yu, Yixuan Jiang, Chengguang Tang, Yongbin Li, and Jian Sun. 2020. A Survey on Dialog Management: Recent Advances and Challenges. *arXiv preprint arXiv:2005.02233* (2020).
[10] 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: Human Language Technologies*. 4171–4186.
[11] Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. (2019).
[12] Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural Approaches to Conversational AI. *Foundations and Trends® in Information Retrieval* 13, 2-3 (2019), 127–298. https://doi.org/10.1561/1500000074
[13] Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin, and Dilek Hakkani-Tur. 2020. From Machine Reading Comprehension to Dialogue State Tracking: Bridging the Gap. *arXiv preprint arXiv:2004.05827* (2020).
[14] Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, Dilek Hakkani-Tur, and Amazon Alexa AI. 2019. Dialog State Tracking: A Neural Reading Comprehension Approach. In *20th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 264–273.
[15] Pavel Gulyaev, Eugenia Elistratova, Vasily Konovalov, Yuri Kuratov, Leonid Pugachev, and Mikhail Burtsev. 2020. Goal-oriented multi-task bert-based dialogue state tracker. *arXiv preprint arXiv:2002.02450* (2020).
[16] Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-End Neural Pipeline for Goal-Oriented Dialogue Systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 583–592.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[18] Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. *arXiv preprint arXiv:2005.02877* (2020).
[19] Matthew Henderson. 2015. Machine learning for dialog state tracking: A review. (2015).
[20] Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. 292–299.
[21] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796* (2020).
[22] Jiaying Hu, Yan Yang, Chencai Chen, Zhou Yu, et al. 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*. 6366–6375.
[23] 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*. 7109–7118.
[24] 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*. 567–582. https://doi.org/10.18653/v1/2020.acl-main.53
[25] 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*. 567–582.
[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[27] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5478–5483.
[28] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
[29] Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. MinTL: Minimalist Transfer Learning for Task-Oriented Dialogue Systems. *arXiv preprint arXiv:2009.12005* (2020).
[30] Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. 2019. An end-to-end dialogue state tracking system with machine reading comprehension and wide & deep classification. *arXiv preprint arXiv:1912.09297* (2019).
[31] Xiangyang Mou, Brandyn Sigouin, Ian Steenstra, and Hui Su. 2020. Multimodal Dialogue State Tracking By QA Approach with Data Augmentation. *arXiv preprint arXiv:2007.09903* (2020).

[32] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 794–799.

[33] Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural Belief Tracker: Data-Driven Dialogue State Tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1777–1788.

[34] Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899* (2018).

[35] Yawen Ouyang, Moxin Chen, Xinyu Dai, Yinggong Zhao, Shujian Huang, and CHEN Jiajun. 2020. Dialogue State Tracking with Explicit Slot Connection Modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 34–40.

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

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[38] Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. 2018. Multi-task Learning for Joint Language Understanding and Dialogue State Tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. 376–384.

[39] Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 561–568.

[40] Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. 2019. Scaling multi-domain dialogue state tracking via query reformulation. *arXiv preprint arXiv:1903.05164* (2019).

[41] Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and Accurate Dialogue State Tracking via Hierarchical Sequence Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 1876–1885.

[42] Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards Universal Dialogue State Tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2780–2786.

[43] 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*. 6322–6333.

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[45] Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language* 24, 4 (2010), 562–588.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[47] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[48] Dingmin Wang, Chenghua Lin, Li Zhong, and Kam-Fai Wong. 2020. Dialogue State Tracking with Pretrained Encoder for Multi-domain Trask-oriented Dialogue Systems. *arXiv preprint arXiv:2004.10663* (2020).

[49] Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*. 423–432.

[50] TH Wen, D Vandyke, N Mrkšíc, M Gašíc, LM Rojas-Barahona, PH Su, S Ultes, and S Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference*, Vol. 1. 438–449.

[51] Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse* 7, 3 (2016), 4–33.

[52] Jason D Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. 282–291.

[53] Jason D Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language* 21, 2 (2007), 393–422.

[54] Chien-Sheng Wu, Steven CH Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 917–929.

[55] 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. *arXiv preprint arXiv:1905.08743* (2019).

[56] Puyang Xu and Qi Hu. 2018. An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 1448–1457.

[57] Puhai Yang, Heyan Huang, and Xian-Ling Mao. 2020. Context-Sensitive Generation Network for Handing Unknown Slot Values in Dialogue State Tracking. *arXiv preprint arXiv:2005.03923* (2020).

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

[59] Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 1458–1467.

[60] Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020. Efficient Context and Schema Fusion Networks for Multi-Domain Dialogue State Tracking. *arXiv preprint arXiv:2004.03386* (2020).

[61] Lukas Zilka and Filip Jurcicek. 2015. Incremental LSTM-based dialog state tracker. In *2015 Ieee Workshop on Automatic Speech Recognition and Understanding (Asru)*. IEEE, 757–762.

A SLOT-SPECIFIC ACCURACY

Table 6: Slot-specific accuracy (%) computed via considering all dialogues in the test set of MultiWOZ 2.1 and only the dialogues that contain the domain the slot belongs to (in gray).

Slot	SOM-DST	TripPy	STAR	SOM-DST	TripPy	STAR
attraction-area	97.08	96.64	**97.76**	92.02	90.15	**93.80**
attraction-name	94.46	**97.16**	94.20	83.44	**91.67**	82.42
attraction-type	96.42	96.16	**96.68**	89.39	88.63	**90.94**
hotel-area	**95.93**	95.53	95.81	88.40	87.29	**88.54**
hotel-book day	99.14	99.14	**99.40**	97.56	97.56	**98.29**
hotel-book people	99.14	98.90	**99.41**	97.56	96.87	**98.32**
hotel-book stay	99.21	99.25	**99.40**	97.75	97.95	**98.28**
hotel-internet	96.76	**96.97**	96.93	90.76	91.38	**91.42**
hotel-name	94.34	**97.38**	95.20	84.19	**92.82**	86.31
hotel-parking	96.95	97.34	**97.51**	91.34	92.42	**92.89**
hotel-pricerange	97.00	96.86	**97.15**	91.61	91.23	**92.37**
hotel-stars	97.87	98.03	**98.08**	93.93	94.40	**94.63**
hotel-type	93.68	93.19	**94.51**	82.14	80.76	**84.37**
restaurant-area	96.81	96.56	**97.39**	92.60	91.67	**93.67**
restaurant-book day	99.36	99.07	**99.39**	98.40	97.67	**98.43**
restaurant-book people	99.13	98.73	**99.38**	97.81	96.81	**98.71**
restaurant-book time	98.82	99.00	**99.43**	97.01	97.50	**98.59**
restaurant-food	97.22	97.20	**97.78**	93.06	93.03	**94.34**
restaurant-name	92.78	**96.92**	93.68	82.13	**92.40**	83.84
restaurant-pricerange	97.34	97.34	**97.51**	93.89	93.86	**94.61**
taxi-arriveby	99.09	99.06	**99.28**	90.65	90.03	**92.07**
taxi-departure	97.75	96.31	**98.49**	76.63	59.35	**83.81**
taxi-destination	98.26	96.89	**98.61**	83.17	66.98	**85.32**
taxi-leaveat	99.05	**99.26**	98.90	89.40	**91.90**	87.54
train-arriveby	96.65	**97.30**	96.85	91.75	**93.28**	92.14
train-book people	97.41	97.39	**97.95**	93.52	93.49	**94.85**
train-day	98.72	99.17	**99.63**	96.98	98.06	**99.25**
train-departure	98.26	98.01	**98.83**	95.79	95.35	**97.26**
train-destination	**98.75**	98.53	**98.75**	97.11	96.54	**97.42**
train-leaveat	94.30	**95.57**	94.28	85.79	**88.98**	85.69

Traditionally, the slot-specific accuracy for each slot is computed based on all the dialogues in the test set [18, 22, 25, 27, 43]. However, for a given slot, there are many dialogues that do not contain the domain the slot belongs to. In these dialogues, the value of the slot

Table 7: Dialogue state outputs of TripPy and our model STAR on a dialogue (dialogue_id: MUL2686.json) from the test set of MultiWOZ 2.1. For simplicity, only the turn-active dialogue states are presented. The false predictions are shown in blue.

User: Hi! Can you help me find some places to go on the south side of the city? I am super bored! **GroundTruth:** *attraction-area=south* **TripPy:** *attraction-area=south* **STAR:** *attraction-area=south*
Sys: I am happy to help. There are several types of places to go in the south, parks, theatre, cinema, museum, nightclub. What sounds good to you? **User:** I do not have a preference. Could you make a recommendation? **GroundTruth:** *attraction-type=dontcare* **TripPy:** *attraction-type=none* **STAR:** *attraction-type=dontcare*
Sys: How about a nightclub or cinema? There's also a great museum. **User:** Anything is fine, whatever is most popular. **TripPy:** *attraction-type=dontcare*
Sys: Sure, cineworld cinema will be great. What information do you need about this attraction? **User:** Can I get the address, entrance fee and attraction type please? **GroundTruth:** *attraction-name=cineworld cinema* **TripPy:** *attraction-name=cineworld cinema* **STAR:** *attraction-name=cineworld cinema*
Sys: The address is cambridge leisure park, clifton way. It is a cinema. I do not know the entrance fee, unfortunately. **User:** I am also wanting some information on a hotel called the kirkwood house. Can you tell me about them? **GroundTruth:** *hotel-name=kirkwood house* **TripPy:** *hotel-name=kirkwood house* **STAR:** *hotel-name=kirkwood house*
Sys: That is a guest house found in the north. It is in the moderate price range and is rated 4 stars. Do you need their contact details? **User:** I need a booking there for 5 people staying 3 nights from Monday. **GroundTruth:** *hotel-book people=5; hotel-book stay=3; hotel-book day=monday* **TripPy:** *hotel-book people=5; hotel-book stay=3; hotel-book day=monday* **STAR:** *hotel-book people=5; hotel-book stay=3; hotel-book day=monday*
Sys: Excellent. You are all booked. Reference number is j0vgu1ux. Can I help with anything else? **User:** Yes. I need a taxi to commute between the 2 places. I would like to leave the park by 01:15. **GroundTruth**: *taxi-leaveat=01:15; taxi-departure=cineworld cinema; taxi-destination=kirkwood house* **TripPy:** *taxi-leaveat=01*; *taxi-departure=kirkwood house*; *taxi-destination=cineworld cinema* **STAR:** *taxi-leaveat=01:15; taxi-departure=cineworld cinema; taxi-destination=kirkwood house*

is always none. Due to this, even if a model predicts the values of a slot for all dialogues as none, it can still achieve a relatively high slot-specific accuracy. To overcome this limitation, we propose calculating the slot-specific accuracy using only the dialogues that involve the domain the slot belongs to. The detailed results are shown in Table 6 (in gray). For comparison, we also include the results computed based on all dialogues. As can be seen, no matter which method is adopted to calculate the slot-specific accuracy, our model is able to achieve better performance for most slots. Table 6 also shows that if the traditional method is adopted, all three models demonstrate higher than 90% slot-specific accuracy for each slot. Besides, there are only subtle differences in the slot-specific accuracy of the three models. While the slot-specific accuracy computed using our proposed method is more discriminative.

B CASE STUDY

Table 7 shows the predicted dialogue states of our model and TripPy on a dialogue from the test set of MultiWOZ 2.1. As can be seen, our model correctly predicts all slot values at each turn. However, TripPy fails to predict the value of slot "*attraction-type*" at turn 2 and delays the prediction to turn 3. At the last turn, TripPy predicts the value of slot "*taxi-leaveat*" as "*01*" rather than "*01:15*", albeit this information is explicitly contained in the user utterance. For slot "*taxi-departure*" and slot "*taxi-destination*", since the user provides the corresponding information indirectly, it is challenging to deduce their valid values. TripPy falsely predicts the destination as the departure and vice versa.