Slot Attention with Value Normalization for Multi-Domain Dialogue State Tracking

Yexiang Wang, Yi Guo*, and Siqi Zhu
East China University of Science and Technology
wyxiang@mail.ecust.edu.cn, guoyi@ecust.edu.cn
zhusiqi@mail.ecust.edu.cn

Abstract

Incompleteness of domain ontology and unavailability of some values are two inevitable problems of dialogue state tracking (DST). Existing approaches generally fall into two extremes: choosing models without ontology or embedding ontology in models leading to over-dependence. In this paper, we propose a new architecture to cleverly exploit ontology, which consists of Slot Attention (SA) and Value Normalization (VN), referred to as SAVN. Moreover, we supplement the annotation of supporting span for MultiWOZ 2.1, which is the shortest span in utterances to support the labeled value. SA shares knowledge between slots and utterances and only needs a simple structure to predict the supporting span. VN is designed specifically for the use of ontology, which can convert supporting spans to the values. Empirical results demonstrate that SAVN achieves the state-of-the-art joint accuracy of 54.52% on MultiWOZ 2.0 and 54.86% on MultiWOZ 2.1. Besides, we evaluate VN with incomplete ontology. The results show that even if only 30% ontology is used, VN can also contribute to our model.

1 Introduction

Dialogue state tracking (DST) is a core component in the pipeline-based task-oriented dialog systems. The goal of DST is to extract the dialogue states which are indicated by a set of *(domain, slot, value)* triples during conversation. The *(domain, slot, value)* triple represents that previous conversation involves the *slot* of the *domain* and the specific content is the *value*. For example, as shown in Figure 1, *(restaurant, price, expensive)* triple means that user wants to reserve an expensive restaurant. A high-quality DST model plays a significant role in the dialogue system, because the dialogue states determine the next system action (Chen et al., 2017).

Traditional DST approaches generally rely on ontology already defined, where all slots and their possible values are given. With a predefined ontology, DST is simplified to a classification problem. The goal is to choose the most appropriate value from ontology for the slot (Mrkšić et al., 2017; Zhong et al., 2018). However, in practical applications, a complete ontology is almost impossible to be defined in advance. To overcome the drawback, span-based (Xu and Hu, 2018; Gao et al., 2019) and generation (Wu et al., 2019) approaches spring up.

The second problem is that some values required by DST cannot be found in utterances due to the diverse descriptions during a conversation. As shown in Figure 1, value *expensive* was expressed as *high end* in the first turn. The problem gives rise to the powerlessness of span-based approaches. Recently, Zhang et al. (2019) show a dual strategy that combines the advantages of both the picklist-based and span-based methods. They use ontology in span-based approaches to deal with the problem and achieve the SOTA performance, which also shows that the ontology is powerful.

Budzianowski et al. (2018) introduced a large-scale multi-turn dialogue dataset (MultiWOZ) spanning over several domains and topics. As shown in Figure 1, the user initially wants to make a restaurant reservation, then requests information about attractions close to the restaurant, and finally books a taxi. During the conversation, the models for DST should determine whether each *(domain, slot)* pair has a value in each turn to obtain the most relevant *(domain, slot, value)* triples. However, unlike single domain DST problems, in which only a few slots need to be tracked, such as four slots in WOZ (Wen et al., 2017), there are a total of 30 *(domain, slot)* pairs of five domains in MultiWOZ, which

*Corresponding Author.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 3019–3028,
November 16–20, 2020.

User	System	Dialogue state
I would like to find a high end restaurant in the center of the city.		(restaurant, price, expensive) (restaurant, area, centre)
	There is an expensive restaurant called the Ugly Duckling.	(restaurant, price, expensive), …
Can i get a reservation for 7 at 14:00 this coming Friday?		(restaurant, name, the Ugly Ducking), (restaurant, people, 7), (restaurant, time, 14:00), (restaurant, day, Friday)
	I have successfully booked your reservation. Your reference number is ynceb914. Will this be all?	(restaurant, price, expensive), …
I am also looking for some entertainment close to the restaurant.		(attraction, area, centre)
	Is there any type of attraction you would like me to search?	(restaurant, price, expensive), …
Why do not you try an architectural attraction.		(attraction, type, architectural)
	All Saints Church looks good , would you like to head there?	(restaurant, price, expensive), …
That sounds good.		(attraction, name, All Saints Church)
	Is there anything else I can help you?	(restaurant, price, expensive), …
I also need to book a taxi between the restaurant and the church.		(taxi, departure, the Ugly Ducking), (taxi, destination, All Saints Church)
	What time would you like the taxi from the Ugly Ducking?	(restaurant, price, expensive), (restaurant, area, centre), (restaurant, name, the Ugly Ducking), (restaurant, people, 7), (restaurant, time, 14:00), (restaurant, day, Friday), (attraction, area, centre), (attraction, name, All Saints Church), (taxi, departure, the Ugly Ducking), (taxi, destination, All Saints Church), (taxi, leaveAt, 20:00)
20:00, please.		

Figure 1: An example of dialogue state tracking in a conversation. Each turn contains a user utterance (left) and a system utterance (right). The blue words are supporting spans in the utterances and new *(domain, slot, value)* triples in the Dialogue state. In each turn, the DST models need to track slot values mentioned by the user for all the *(domain, slot)* pairs.

can be more in practical applications. Therefore, it requires DST models should determine the slots efficiently.

To tackle these challenges, we emphasize that DST models should optimize the structure of slots determination and utilize ontology more flexibly rather than abandon it. In this paper, we propose to divide the model of DST into Slot Attention (SA) and Value Normalization (VN). Simple and efficient processing of slots and flexible use of ontology are the main advantages of SAVN. Contributions in this work are summarized as [†]:

- SA shares knowledge between slots and utterances and is able to optimize the determination of all slots jointly. Compared to the span-based approach in DS-DST (Zhang et al., 2019), SA improves efficiency by nearly *count(slots)* times in determining the slots.
- Considering that the number of possible slot values in ontology could be large in the real scenario, VN is designed as a simple, flexible, and effective model to use an ontology, which only needs 8 minutes for training on a V100 GPU. VN can choose to directly output the supporting span from SA or select a value in the ontology.
- We supplement the annotation of supporting span for labeled values unavailable in utterances on MultiWOZ 2.1, which can help the span-based model learn semantics more fully and help ontology be better utilized.
- We fully evaluate VN with incomplete ontology. The results show that VN can gain positive performance for SAVN as long as the integrity of ontology is more than 30%. And as we expected, the more complete ontology is, the more VN can rely on it.

[†]The code will be released at https://github.com/wyxlzsq/savn.

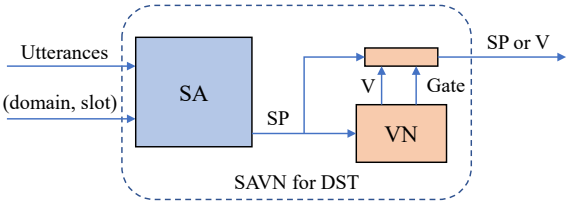

Figure 2: The overview of Slot Attention with Value Normalization for Dialogue State Tracking.

2 SAVN model

The overview of the model is shown in Figure 2, which consists of Slot Attention and Value Normalization. SA outputs the **S**upporting s**P**an (SP) from utterances for each *(domain, slot)* pair. And VN chooses to output supporting span directly or convert supporting span to the *value* in ontology according to the gate.

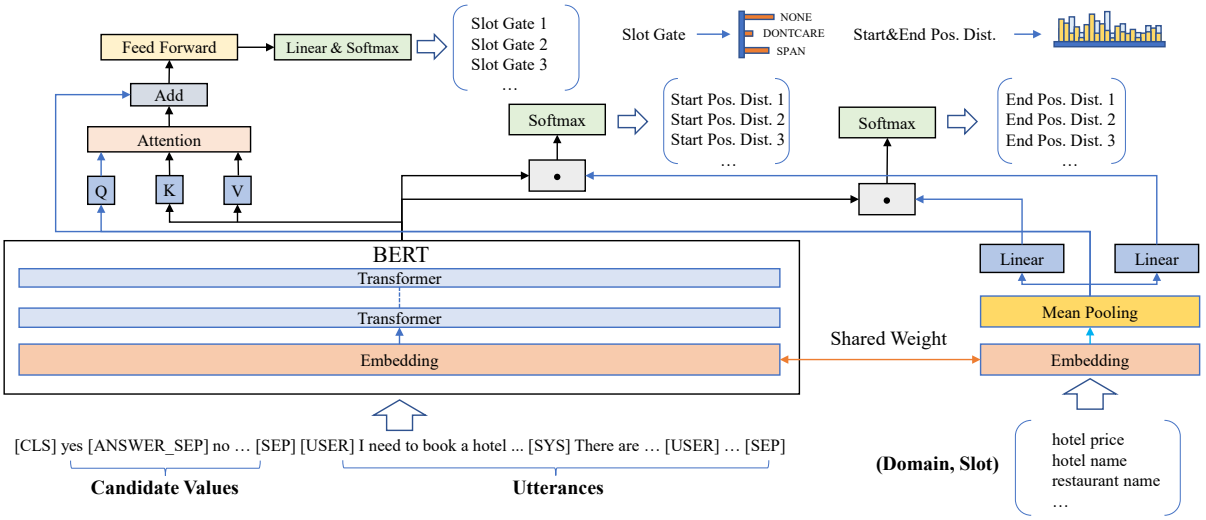

Figure 3: The model architecture of Slot Attention.

2.1 Slot Attention

As shown in Figure 3, Slot Attention (SA) accepts two inputs, one of which is the text of the previous conversations, and the other is a list of *(domain, slot)* pairs. Similar to DS-DST model (Zhang et al., 2019), we also employ BERT (Devlin et al., 2019) as the encoder for utterances. The difference is that we separate slots and utterances and share knowledge between them. Then we directly use the inner product to predict the span in utterances and use an attention module to interact with slots and utterances to classify. Benefiting from this structure, our model can determine the slots in parallel, optimize the determination jointly, and only needs to encode the utterances once for each turn while DS-DST needs to encode *count(slots)* times. Additionally, for SA to have the ability to output some special words, we added some fixed candidate values in front of the utterances such as *yes*, *no*.

Let us define $X = \{(u_1, r_1), ..., (u_n, r_n)\}$ as the set of user utterances and system responses in a conversation with n turns, $C = [a_1, a_2, ..., a_k]$ as the list of k fixed candidate values, and $S = [s_1, s_2, ..., s_j]$ as the list of j *(domain, slot)* pairs. Due to the limitation of the maximum sequence length of BERT, sometimes it is not possible to encode all utterances. Therefore, we set a parameter m to limit the number of turns entered. The input utterances for turn t should be :

$$X_t^m = \begin{cases} [U_1, ..., U_{t-1}, u_t] & \text{if } t \leq m \\ [U_{t-m+1}, ..., u_t] & \text{otherwise,} \end{cases} \quad (1)$$

where U_1 represents $u_1 \oplus r_1$, the $\oplus$ means to concatenate the utterances of u_1 and r_1. r_t is the system response of turn t, so $r_t \notin X_t^m$.

Then by encoding the utterances of turn t by BERT and embedding the slots by the Embedding module of BERT, we can get the hidden states of utterances and slots as follows:

$$\begin{aligned} I &= C \oplus X_t^m, \\ H_t^u &= \text{BERT}(I), \\ E_t^s &= \text{Embedding}(S), \\ H_t^s &= \text{MeanPooling}(E_t^s), \end{aligned} \quad (2)$$

where $H_t^u \in \mathbb{R}^{p \times h}$ and $H_t^s \in \mathbb{R}^{q \times h}$. p is the sequence length of I, q is the number of *(domain, slot)* pairs and h is the dimension of the BERT hidden state.

2.1.1 Slot Gate Classification

As introduced in Section 1, There are many *(domain, slot)* pairs in Multi-domain DST problem, which make it more challenging than single-domain DST problem. Similar to TRADE model (Wu et al., 2019), we design a classification module with *none*, *dontcare* and *span* as a slot gate. For each *(domain, slot)* pair, if the slot gate predicts *none* or *dontcare*, we ignore the span predicted from utterances and fill the pair with "none" or "do not care".

The module to classify slots is similar to a Transformer (Vaswani et al., 2017) block. We employ the "Scaled Dot-Product Attention" to get an

utterances-aware slot representation A_u^s:

$$
\begin{aligned}
Q_t &= H_t^s \cdot W_q + b_q, \\
K_t &= H_t^u \cdot W_k + b_k, \\
V_t &= H_t^u \cdot W_v + b_v, \\
A_u^s &= \text{Softmax}(\frac{Q_t {K_t}^T}{\sqrt{d_k}}) V_t,
\end{aligned} \tag{3}
$$

where $A_u^s \in \mathbb{R}^{q \times h}$.

Then in order to better integrate the states of slots and utterances, we add A_u^s and H_t^s to get H_c as the features to classify:

$$
\begin{aligned}
H_c' &= \text{GELU}(H_c \cdot W_c + b_c), \\
G_t &= \text{Softmax}(H_c' \cdot W_l + b_l) \in \mathbb{R}^{q \times 3},
\end{aligned} \tag{4}
$$

where G_t is the slot gates of all *(domain, slot)* pairs at turn t.

2.1.2 Span-Based Value Prediction

For each *(domain, slot)* triple, span-based methods obtain the *value* by predicting a span with start and end position in utterances. In order to make the slot determination more efficient, we simplify the structure of span-based predictions. We can get the span predictions by:

$$
\begin{aligned}
D_t^s &= H_t^s \cdot W_s + b_s, \\
D_t^e &= H_t^s \cdot W_e + b_e, \\
P_t^s &= \text{Softmax}(D_t^s \cdot (H_t^u)^T) \in \mathbb{R}^{q \times p}, \\
P_t^e &= \text{Softmax}(D_t^e \cdot (H_t^u)^T) \in \mathbb{R}^{q \times p},
\end{aligned} \tag{5}
$$

where P_t^s and P_t^e are the start position distributions and end position distributions of all *(domain, slot)* pairs at turn t respectively.

2.1.3 Optimization

We can optimize all slots determination jointly. The joint losses at turn t are as follows:

$$
\begin{aligned}
L_g &= \sum_{q=1}^{Q} -\log(G_q \cdot (y_q^g)^T), \\
L_s &= \sum_{q=1}^{Q} -\log(P_q^s \cdot (y_q^s)^T),
\end{aligned} \tag{6}
$$

where L_g is the loss of the slot gate predictions, L_s and L_e are the loss of the start and end position predictions respectively. And Q is the number of *(domain, slot)* pairs, y is the true one-hot label.

Similar to L_s, we can get the end loss L_e. Then we optimize the weighted-sum of these three loss functions using hyper-parameters α and β,

$$
L = \alpha L_g + \beta(L_s + L_e). \tag{7}
$$

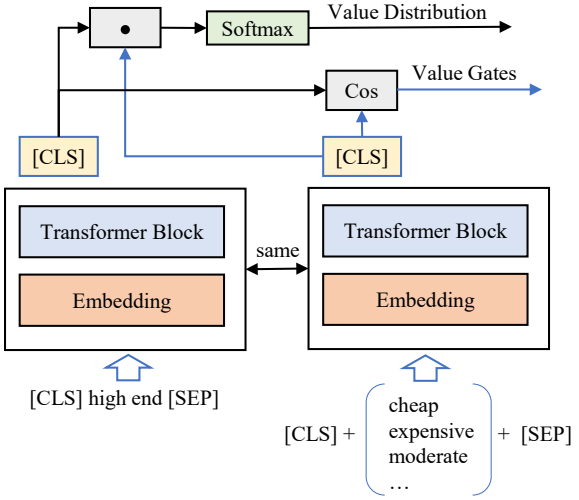

Figure 4: The model architecture of Value Normalization.

2.2 Value Normalization

Value Normalization is a flexible module for utilizing ontology, which can also be combined with other DST models. Considering that there are numerous possible values in the ontology and few data for training, we design a simple and effective model for VN, which can also benefit from the pre-trained BERT model.

As shown in Figure 4, VN is designed with one layer of the transformer block, which we call VN^1. By analogy, we can get VN^4 and VN^{12} (i.e., use BERT-base model as encoder). The model will load parameters from the corresponding layers of BERT. In Section 4.2, the experimental results show that VN^1 has done well enough for the MultiWOZ dataset.

Let us define T_0 is the hidden state of the first token ([CLS]) after transformer. Then we use $T_0^s \in \mathbb{R}^h$ as the hidden state of the supporting span after encoding and $T_0^o \in \mathbb{R}^{n \times h}$ as the hidden states of n possible values in ontology for the corresponding *(domain, slot)* pair. We use the inner products of the supporting span and the possible values as the matching scores, which is defined as:

$$
M = \text{Softmax}(T_0^s \cdot (T_0^o)^T) \in \mathbb{R}^n. \tag{8}
$$

Then we can get the max matching value. In addition, we employ the cosine similarity as the value gate since the ontology may be incomplete.

$$
cos(T_0^{V^m}, T_0^s) = \frac{T_0^{V^m} \cdot (T_0^s)^T}{\|T_0^{V^m}\| \|(T_0^s)^T\|}, \tag{9}
$$

	Labeled Value	Supporting Span
Varied Expressions	expensive	high end
	0	no stars
	09:15	after 9am
	cinema	movie theaters
Spelling Mistakes	west	wets
	thursday	thirsday
	expensive	expensiove
	museum	musems
Annotation Errors	1145	11:45
	sunday	saturday
	08:15	18:15
	5	4

Table 1: Some examples of supporting span annotation.

the final output is:

$$\text{output} = \begin{cases} V^m & cos(T_0^{V^m}, T_0^s) > \theta \\ \text{SP} & cos(T_0^{V^m}, T_0^s) \leq \theta, \end{cases} \quad (10)$$

where V^m is the max matching value, SP is the supporting span and θ is a hyper-parameter.

Our loss function for optimizing VN is defined as follows:

$$L_i = \begin{cases} -\log(M \cdot (y^v)^T) & r = 1 \\ \max(cos(T_0^o, T_0^s)) + 1 & r = 0, \end{cases} \quad (11)$$

where y^v is the true one-hot label and $r = 1$ means the value for the supporting span is in ontology. However, r will always be equal to 1 without preprocessing in training because ontology is invariably complete for the training set. In our experiments, we employ full training set to train VN with incomplete ontology in order to get dispersed vector representations for values.

3 Annotation for Supporting Span

Our annotation work is based on the MultiWOZ 2.1 dataset (Eric et al., 2019), which is a fixed version of the MultiWOZ 2.0 dataset (Budzianowski et al., 2018). MultiWOZ 2.1 dataset is a large-scale collection of human-human written conversations over multiple domains and topics, which has labeled 63,662 *(conversation, domain, slot, value)* quadruples (except "none" value) in the training set.

Annotation for supporting span is mainly to address the problem that some labeled value can not be found in the conversations. The causes of this problem can be divided into three categories: varied expressions, spelling mistakes, and annotation errors.

The criterion of annotations is to find the shortest span in the conversations, which can help us get the labeled value. Based on the criterion, we annotate 936 *(supporting span, value)* pairs on MultiWOZ 2.1 training set, in which varied expressions account for 637 (68%), spelling mistakes account for 123 (13%), and annotation errors account for 176 (19%). Table 1 shows some examples of supporting span annotation.

After annotation, we can change *(domain, slot, value)* triples in training set to *(domain, slot, supporting span, value)* quadruples, where the *supporting span* will be equal to the *value* if the *value* can be found in the conversations. Then we employ *(domain, slot, supporting span)* triples to train SA and *(supporting span, value)* pairs to train VN. Specifically, we do not use the annotation of annotation errors to train VN, for it should not convert Saturday to Sunday.

4 Experiments

We evaluate our model on two publicly available datasets: MultiWOZ 2.0 and MultiWOZ 2.1, both of which are fully-labeled task-oriented corpora comprised of human-human written conversations and contain 8,438 multi-turn dialogues with each dialogue having 13.68 turns on average in training set (Budzianowski et al., 2018). The difference between MultiWOZ 2.0 and MultiWOZ 2.1 is that MultiWOZ 2.1 has changed more than 32% of state annotations across 40% of the dialogue turns to fix the noisy state annotations in MultiWOZ 2.0 (Eric et al., 2019).

Following previous work (Wu et al., 2019), only five domains (i.e., restaurant, hotel, attraction, taxi, and train) are employed in our experiments because the dialogues that belong to the other two domains (i.e., hospital and police) are rare in the training set and do not appear in the test set. As introduced in Section 3, we get a new training set by using the supporting span annotations, which can be called the SP training set. Additionally, there are no changes to the test set and the dev set.

4.1 Training Details

We use the pre-trained BERT-base-uncased model as the utterance encoder in SA, which has 12 hid-

	Joint	Slot
GLAD	35.57	95.44
Neural Reading	41.10	-
SUMBT	46.65	96.44
TRADE	48.62	96.92
DSTQA	51.44	97.24
SA_{raw}	48.26	97.07
+ VN^1	52.84 (+4.58)	97.34
+ VN^{12}	53.04 (+4.78)	97.35
SA_{sp}	48.44	97.02
+ VN^1	54.36 (+5.92)	97.41
+ VN^{12}	**54.52 (+6.08)**	**97.42**

Table 2: Results on MultiWOZ 2.0 dataset.

	Joint	Slot
DST-Span	40.39	-
TRADE	45.60	-
DSTQA	51.17	97.21
DS-DST	51.21	-
DST-Picklist	53.30	-
SA_{raw}	45.72	96.89
+ VN^1	50.76 (+5.04)	97.24
+ VN^{12}	50.73 (+5.01)	97.24
SA_{sp}	45.74	96.90
+ VN^1	**54.86 (+9.12)**	**97.55**
+ VN^{12}	54.80 (+9.06)	97.55

Table 3: Results on MultiWOZ 2.1 dataset.

den layers with 768 units. For the limitation of the maximum sequence length, We set m (in equation 1) to 9. If the current conversation turn exceeds m, we will combine the predicted dialogue states with the previous dialogue states to complete dialogue states for the current turn.

In our experiments, SA and VN are both trained with Adam optimizer (Kingma and Ba, 2014) in which the learning rate linearly decreases from $5e^{-5}$ and $1e^{-4}$, respectively. We have trained SA with 3 epochs and VN with 5 epochs both on MultiWOZ 2.0 and MultiWOZ 2.1. Specifically for VN, we train VN^1 and VN^{12} (introduced in Section 2.2) to compare their performance.

Our results can be reproduced with a 16 GB V100 GPU in 2 hours (8 minutes for VN^1).

4.2 Results

Two standard metrics, joint accuracy and slot accuracy, can be employed to evaluate the performance of our model. Joint accuracy is the accuracy of dialogue states, which requires that all *(domain, slot, value)* triples in the dialogue states are predicted correctly. And slot accuracy is the accuracy of *(domain, slot, value)* triples, which requires that the predicted value of *(domain, slot)* pair is predicted correctly. The joint accuracy is a more challenging metric, for there is a considerable number of *(domain, slot)* pairs in dialogue states.

To better evaluate the role of supporting spans, we have trained two versions of SA, one of which utilizes the original training set called SA_{raw} and the other employs the SP training set called SA_{sp}. And we make a comparison with the following existing models:

- GLAD (Zhong et al., 2018) shares parameters among slots by virtue of global modules and applies the local modules to learn slot-specific features.
- Neural Reading (Gao et al., 2019) formulates DST as a reading comprehension task. The model encodes the word tokens by a pre-trained BERT model, then obtains the contextual representation by LSTM.
- SUMBT (Lee et al., 2019) learns the slot and utterance representations by fine-tuning a pre-trained BERT model. Then they compute the similarity between possible values and utterances via a slot-utterance matching module.
- TRADE (Wu et al., 2019) employs an encoder-decoder architecture to generate the values for slots from the vocabulary and the dialogue history.
- DSTQA (Zhou and Small, 2019) models DST as a question answering problem, which generates a question to ask for the value of the slot at each turn.
- DS-DST (Zhang et al., 2019) proposes a Dual Strategy to combine the advantages of the picklist-based and span-based methods, which has been evaluated individually as DST-Picklist and DST-Span.

On MultiWOZ 2.0, as shown in Table 2, our model achieves the highest performance, 54.52% of joint accuracy in which VN gains 6.08% absolute improvement. And on MultiWOZ 2.1, as shown in Table 3, our model also achieves the highest performance, 54.86% of joint accuracy in which

θ	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
-1.0	8.41	12.05	13.19	14.67	19.46	22.41	31.49	38.50	46.76	**54.86**
-0.5	33.35	22.30	17.48	20.15	20.89	26.97	31.84	38.68	46.76	54.86
0	33.35	25.83	25.99	24.86	22.58	29.61	36.43	41.67	48.29	54.86
0.3	33.35	30.20	33.66	38.18	39.81	48.60	48.75	51.43	53.5	54.86
0.5	33.44	35.71	43.44	46.88	**49.43**	**49.81**	**51.40**	**51.87**	**53.57**	54.86
0.7	34.03	40.64	**45.97**	**47.19**	48.14	49.42	51.28	49.55	51.18	54.86
0.9	33.44	41.76	45.39	46.10	46.90	47.84	48.19	46.52	47.43	52.10
1.0	**45.74**	**45.74**	45.74	45.74	45.74	45.74	45.74	45.74	45.74	45.74

Table 4: The performance of SAVN with incomplete ontology on MultiWOZ 2.1. The percentage in the header refers to the usage rate of ontology and the θ is introduced in Equation 10.

VN gains 9.12% absolute improvement. Combining results from two tables, we demonstrate that the performance of SA is similar to TRADE and SA has about 5% higher absolute performance than DST-span, which is also a span-based method using BERT.

Comparing SA_{raw} with SA_{sp}, we find that their performance is similar without VN and the improvement of SA_{sp} performance is obviously greater than that of SA_{raw} by VN, which shows that SA_{sp} has learned more about semantics so that it could output the supporting span and can be better combined with VN. Furthermore, by comparing VN^1 with VN^{12}, we prove that VN has enough performance for the MultiWOZ dataset with only one transformer layer.

4.3 Incomplete Ontology

The experimental results in Table 2 and Table 3 show that ontology is a powerful resource for DST. However, it is impractical to get a full ontology in advance when the DST model is oriented to practical applications, which leads some models, such as TRADE, to abandon ontology. In this section, We choose SA_{sp} on MultiWOZ 2.1 as the base model to evaluate the performance of VN^1 with incomplete ontology.

There are many slots in the ontology. We can divide them into two categories, common-value slots and special-value slots. Common-value slots, such as *hotel-price* and *hotel-type*, are able to include all possible values as long as a few values are given. And for special-value slots, such as *restaurant-name* and *taxi-departure*, it is difficult to cover all possible values by predefined values. In our experiment, we only drop out values in special-value slots and always keep all values of common-value slots.

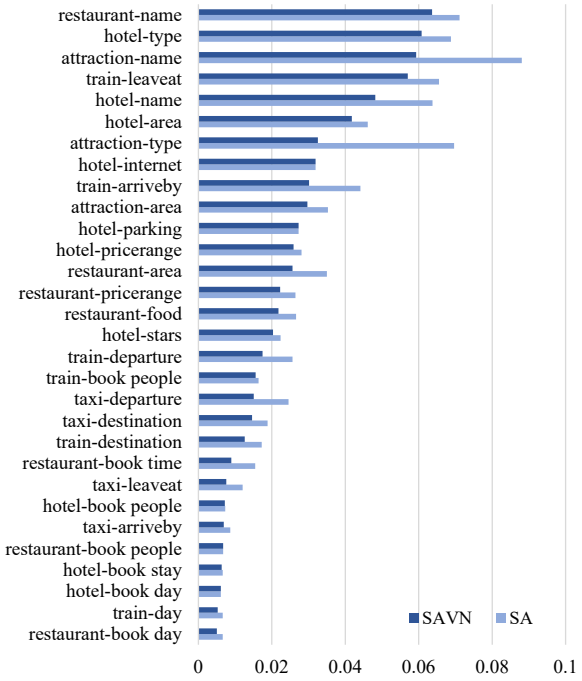

Figure 5: The error rate of slots on MultiWOZ 2.1.

The results are shown in Table 4. Even if there is only 30% ontology, VN can bring positive performance as long as θ is appropriate. Based on the results, we demonstrate that the performance of VN has steadily improved with the increased usage rate of ontology. Furthermore, the more complete ontology is, the smaller θ can be, which means VN can be more dependent on ontology.

5 Error Analysis

An error analysis of SA_{sp} with VN^1 on MultiWOZ 2.1 is shown in Figure 5. The three slots with the highest error rates are *restaurant-name* with 6.37%, *hotel-type* with 6.08% and *attraction-name* with 5.93%. Through the detailed analysis of error samples, we observe many labeled states do not

include the name that only appears in system response. These states are similar to the example in Figure 1 with the restaurant-name and attraction-name removed. Once the difference occurs, it will lead to errors in the subsequent dialogue states, resulting in high error rates. And the labels of *hotel-type* are found to be confusing. For instance, for the sentence "I am looking for a hotel with ...", sometimes the label of *hotel-type* is *hotel* and sometimes it is *none*.

Compared with SA, SAVN has significantly lower error rates on *attraction-type* and *attraction-name*. The improvement of *attraction-name* is mainly due to the repair of spelling mistakes, and the improvement of *attraction-type* mainly benefits from the normalization of varied expressions. It is worth mentioning that VN can not improve the accuracy of some slots, which only need to be filled with *yes* or *no*, such as *hotel-internet* and *hotel-parking*.

6 Related Work

Traditional dialogue state tracking models extract utterance semantics by hand-crafted features and complex domain-specific lexicons (Wang and Lemon, 2013; Williams, 2014; Henderson et al., 2014) to predict the dialogue states, which is hard to adapt to new domains. Then, to overcome this drawback, Mrkšić et al. (2017) propose a novel Neural Belief Tracking (NBT) framework with learning n-gram representation of utterance by using a convolutional neural network, and achieve better performance. At the same time, Models for multi-domain DST have then been proposed. Rastogi et al. (2017) build a multi-domain DST model by two-layer bi-GRU and Ramadan et al. (2018) track domain and the dialogue states jointly through multiple bi-LSTM. They employ semantic similarity between utterances and the values in ontology and allow the knowledge to be shared across domains. To transfer knowledge between slots, Zhong et al. (2018) propose a global-local architecture to share parameters among slots and Ren et al. (2018) propose StateNet that shares all parameters among slots and fix the word embeddings during training to handle new slots.

After the pre-trained BERT model showed superior performance, encoding by BERT has become the mainstream. Lee et al. (2019) encode the slots and utterances with BERT, and then compute the similarity between possible values and utterances after a Multi-head attention layer. And Zhang et al. (2019) also employ BERT to encode the utterances. The difference is that they combine the picklist-based and span-based methods and get higher performance. In order to eliminate the dependence on ontology, Wu et al. (2019) propose an encoder-decoder architecture with a pointer network to generate the value for each slot. And Zhou and Small (2019) formulate multi-domain DST as a question answering problem and learn relationships between slots by a dynamically-evolving knowledge graph. Most recently, Heck et al. (2020) propose to use copy mechanisms to fill slots with values, which combine span-based methods with memory methods to avoid the use of value picklists.

7 Conclusion

We introduce a new architecture that divides the prediction of slots and the use of ontology. SA shares parameters not only among all slots but also between slots and utterances. And VN can handle ontology flexibly with a simple and effective structure, which is able to work with incomplete ontology. Combining SA with VN, SAVN has shown excellent performance on both MultiWOZ 2.0 and MultiWOZ 2.1. And we also introduce the annotation of supporting span. In future work, the supporting span annotation can be added to the datasets of a task-oriented dialog system, for the reason that supporting span serves as a bridge between diverse descriptions of users and the normative values in the system. Furthermore, DST models with supporting span allow for a fairer comparison regardless of whether the ontology is used.

Acknowledgments

This work is financially supported by the National Key Research and Development Program of China (grant number 2018YFC0807105), National Natural Science Foundation of China (grant number 61462073), and Science and Technology Committee of Shanghai Municipality (STCSM) (under grant numbers 17DZ1101003, 18511106602 and 18DZ2252300).

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.

Yun-Nung Chen, Asli Celikyilmaz, and Dilek Hakkani-Tür. 2017. Deep learning for dialogue systems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 8–14, Vancouver, Canada. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines.

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

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.

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)*, pages 292–299, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, Florence, Italy. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Osman Ramadan, Paweł Budzianowski, and Milica Gašić. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 432–437, Melbourne, Australia. Association for Computational Linguistics.

A. Rastogi, D. Hakkani-Tür, and L. Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 561–568.

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*, pages 2780–2786, Brussels, Belgium. Association for Computational Linguistics.

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

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*, pages 423–432, Metz, France. Association for Computational Linguistics.

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

Jason D. Williams. 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)*, pages 282–291, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1457, Melbourne, Australia. Association for Computational Linguistics.

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.

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 (Volume 1: Long Papers)*, pages 1458–1467, Melbourne, Australia. Association for Computational Linguistics.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering.