# SAS: Dialogue State Tracking via Slot Attention and Slot Information Sharing

**Jiaying Hu[1], Yan Yang[1,3]*, Chengcai Chen[2], Liang He[1,3], Zhou Yu[4]**
[1]East China Normal University
[2]Xiaoi Research, Xiaoi Robot Technology Co., Ltd
[3]Shanghai Key Laboratory of Multidimensional Information Processing
[4]University of California, Davis
51184506058@stu.ecnu.edu.cn, {yanyang,lhe}@cs.ecnu.edu.cn
arlenecc@xiaoi.com, joyu@ucdavis.edu

## Abstract

Dialogue state tracker is responsible for inferring user intentions through dialogue history. Previous methods have difficulties in handling dialogues with long interaction context, due to the excessive information. We propose a Dialogue State Tracker with Slot Attention and Slot Information Sharing (SAS) to reduce redundant information's interference and improve long dialogue context tracking. Specially, we first apply a Slot Attention to learn a set of slot-specific features from the original dialogue and then integrate them using a Slot Information Sharing. The sharing improve the models ability to deduce value from related slots. Our model yields a significantly improved performance compared to previous state-of-the-art models on the MultiWOZ dataset.

## 1 Introduction

The recent global adoption of personal assistants such as Alexa and Siri made dialogue system a more popular topic in research. The major difference between dialogue systems and question-answering is that dialogue systems need to track dialogue history effectively. So, we normally use a dialogue state tracking component to track user's intention throughout the conversation. A dialogue state is typically composed as a set of slot value pairs in a task-oriented dialogue, such as "hotel-internet-yes". It means the slot "hotel-internet" has a value of "yes".

Early dialogue state tracking model needs a predefined ontology which means the values of every slot are enumerated in advance (Henderson et al., 2014; Mrkšić et al., 2017; Zhong et al., 2018; Sharma et al., 2019). Such practice is inefficient and costly. The large number of possible slot-value pairs makes deploying these models in the real-life applications difficult (Rastogi et al., 2017). This difficulty is further amplified in multi-domain dialogue state tracking where the dialogues have more than one tasks. Because the manual effort grows exponentially with the complexity of the dialogues. In (Wu et al., 2019), Wu et al. introduced a transferable dialogue state generator (TRADE), which can generate dialogue states from utterances using a copy mechanism. This generative model achieved relative good performance, but it still has trouble in extracting relevant information from the original dialogues. For example, a user may tell the agent that he/she needs a taxi in a turn, but the taxi's departure location is implicitly mentioned several turns ago. Inspired by the (Chen et al., 2017; Chen, 2018), (Chen et al., 2019) studied on utilizing attention mechanism to deal with the long distance slot carryover problem. In their work, they first fused the information of the slot, its corresponding value and the dialogue distance into a single vector. Then they computed the attention between this single vector and the concatenation of dialogue and intent information. We simplify the attention method and introduce it into the dialogue state tracking task.

Moreover, it is a common sense that there is some kind of relevance between two slots involving the same domain or the same attribute. For example, people tend to have a meal near the attraction they visit, so slot "attraction-area" and slot "restaurant-area" have the same value at most times. For these slots with a common or related value, if a slot never or seldom appears in the training set, sharing the learned feature of data-sufficient slot may benefit the model's tracking ability on these rare or unknown slots.

So we propose SAS, a new multi-domain dialogue state tracking model to resolve this issue to some extent. To be specific, we use an Slot Attention to localize the key features from the original information-excessive dialogue and a Slot Infor-

*Corresponding author.



*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6366–6375

mation Sharing to improve the models ability to deduce value from related slots. The processed information provided by the slot attention and the sharing module makes the generator more sensitive to the location of the values in the dialogue history and thus generates correct slot values. Experiments on the multi-domain MultiWOZ dataset (Budzianowski et al., 2018) shows SAS can achieve 51.03% joint goal accuracy and outperform previous state-of-the-art model by 2.41%. On the single domain dataset which only contains the restaurant domain, we achieve 67.34% joint goal accuracy, outperforming prior best by 1.99%. In addition, we conduct an analysis of the experimental results to evaluate the quality of values generated by our model.

## 2 Related Work

The early research of DST focused on the pipelined approach which involves a special module named Spoken Language Understanding (SLU) before the DST module (Wang and Lemon, 2013; Williams, 2014; Perez and Liu, 2017). But obviously, it was not reasonable to train SLU and DST respectively since the accumulated error in SLU may be passed to the DST. In order to alleviates this problem, later study focuses on the joint training methods (Henderson et al., 2014; Zilka and Jurcicek, 2015; Wen et al., 2017). Although the higher performance shows the effectiveness of models without SLU, there still remains some shortcomings. For example, these models typically rely on semantic dictionaries which list the potential rephrasings for all slots and values in advance. Make such a list is costly. Fortunately, the recent development of deep learning and representation learning helps the DST to get rid of this problem. (Mrkšić et al., 2017) proposed a novel Neural Belief Tracking (NBT) framework which was able to learn distributed representations of dialogue context over pre-trained word vectors, while (Dernoncourt et al., 2017) described a novel tracking method which used elaborate string matching and coreference resolution to detect values explicitly presented in the utterance. These models greatly improve the performance of DST, but they are not good at handling rare and unknown slot value pairs which seldom or never appear in the training set.

There were many efforts to exploit general features between rare slot value pairs and common ones. (Zhong et al., 2018) proposed GLAD, a model which built global modules to share parameters between estimators for different slots and local modules to learn slot-specific features. (Nouri and Hosseini-Asl, 2018) improved GLAD by reducing the latency in training and inference time, while preserving its powerful performance of state tracking. But as the dialogues become increasingly complex, the performance of these models on multi-domain is not as satisfying as on single domain. Because of the dependency on the dialogue ontology, they have difficulty in scaling up with domains. Once the number of domains increases, the amount of slot value pairs will boom. With the copy mechanism, the sequence-to-sequence model TRADE (Wu et al., 2019) successfully got rid of any predefined slot value pairs and generated dialogue states from conversation utterances.

But we find there still remain several crucial limitations which have not been well solved on multi-domain dialogues. First, these models rely on the long dialogue history to identify the values which belong to various domains and slots. Sometimes the information contained in the dialogue history is too rich for these models to efficiently utilize and the redundant information tends to interfere with their value identification or value generation. Second, the related information among similar slots is wasted. To alleviate these problems, a slot attention and a slot information sharing module are suggested. The former can isolate the most valuable information for each slot, while the latter integrates information kept by its all similar slots and improve the models ability to deduce value from related slots.

## 3 Task Definition

The dialogue state tracking models take the interaction context as input and extract slot value pairs explicitly or implicitly presented in conversations. The combinations of these slot value pairs are the representations of the user's goal. In this paper, we denote $X = \{(u_1, r_1), \cdots, (u_T, r_T)\}$ as the dialogue history, where $u_1, \cdots, u_T$ and $r_1, \cdots, r_T$ are respectively the set of user utterances and the set of system responses in $T$ turns. The dialogue state of turn $t$ is marked as $ST_t$= (slot: $s_j$, value: $y_j^{value}$). Here, $s_j$ indicates the $j$-th slot, while $y_j^{value}$ means the ground turth value sequence for this slot. All the slots in ontology are obtained by preprocessing the original MultiWOZ dataset with the delexicalization. Moreover, we extend the def-

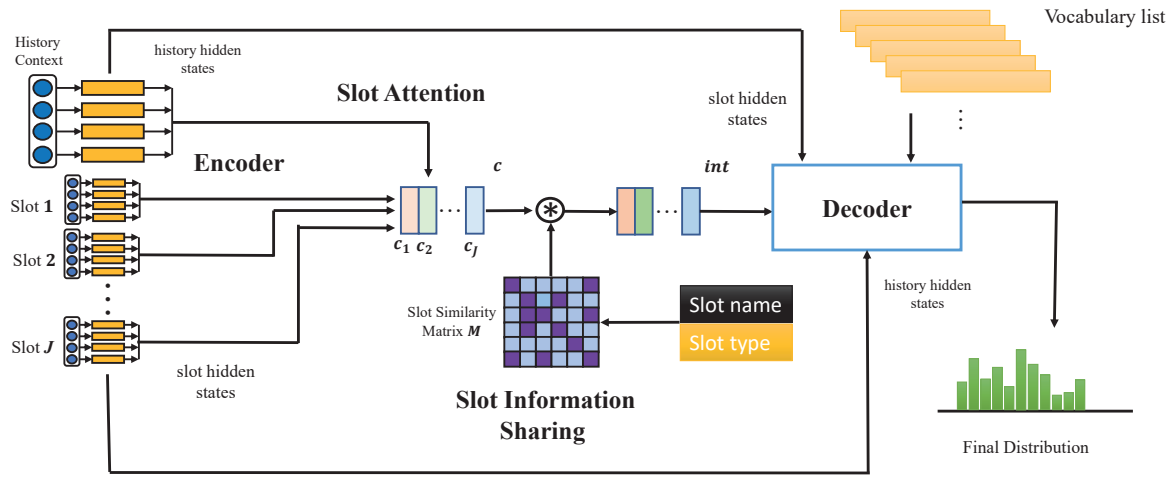


Figure 1: SAS model's architecture. This model consists of four parts: an encoder, a slot attention, a slot information sharing and a decoder.

inition of the slot to include the domain name for convenience. For instance, a slot in this paper will be "hotel-star", rather than "star".

Our primary goal is to learn a generative dialogue state tracker model $M : X \times O \rightarrow ST$ that can efficiently capture the user's intentions for dialogues including multiple domains. And unlike most of the previous models, the ontology $O$ mentioned in this paper only contains the predefined slots and excludes their values.

## 4 Our Proposed Model

Figure 1 shows the architecture of SAS. SAS is a sequence-to-sequence model augmented with slot attention and slot information sharing. Slot attention enables better feature representation and slot information sharing helps understanding less-seen slots. We describe the details of every component in SAS as follows:

### 4.1 Encoder

We use a 1-layer bidirectional gated recurrent unit (GRU) (Chung et al., 2014) to encode the dialogue history. As TRADE (Wu et al., 2019), our input to the model is the concatenation of all words in the recent $l$-turn dialogue history $\mathbf{X_t} = [u_{t-l+1}, r_{t-l+1}, \cdots, u_t, r_t] \in \mathbb{R}^{|X_t| \times d_{emb}}$, where $d_{emb}$ means the embedding size. First, each word in the dialogue history $X$ is mapped to a distributed embedding vector. Then, a GRU is utilized to obtain the hidden state corresponding to each word in the text and we denote these hidden state as the history hidden states $\mathbf{H_t} = \{h_1^{enc}, h_2^{enc}, \cdots, h_{|X_t|}^{enc}\} \in \mathbb{R}^{|X_t| \times d_{hdd}}$.

### 4.2 Slot Attention

To isolate key features from the noisy dialogue history, we build the slot attention. In fact, the multi-domain dialogues are usually complex and contain rich features. This challenges the model's ability to cope with the excessively rich information.

To be specific, in one dialogue, user can mention various information, such as wanting to book a restaurant for a meal and then planning to see an attraction after the meal by ordering a taxi. There are in total 10 slots mentioned spanning across restaurant, attraction and taxi domains. Information from one domain maybe not useful for other domain and can even cause confusion. For example, both restaurant and taxi mention time and people.

So we propose the slot attention to only extract useful history information to every slot. More concretely, for a particular slot $s_j$, we first encode its slot name into slot hidden states $\mathbf{SH_j} = [sh_{j1}^{enc}, \cdots, sh_{j|N|}^{enc}]$, where $|N|$ is the maximum size of the slot name phrase. Since the last hidden state $sh_{j|N|}^{enc}$ provided by the GRU contains the context information of the entire phrase, we pick it as the representation of slot $s_j$.

After that, we calculate the attention between the slot information, $sh_{j|N|}^{enc}$ and the hidden states of the dialogue history $\mathbf{H_t} = [h_1^{enc}, \cdots, h_{|X_t|}^{enc}]$ to obtain the context vector $c_j$:

$$\mathbf{a}^j = (\mathbf{h^{enc}})^\top sh^{enc}_{j_{|N|}} \quad (1)$$

$$sc^j_i = \frac{exp(a^j_i)}{\sum_{i=1}^{|X_t|} exp(a^j_i)} \quad (2)$$

$$c_j = \sum_{i=1}^{|X_t|} sc^j_i h^{enc}_i \quad (3)$$

Here, the score $sc^j_t$ indicates the relevance between info slots $s_j$ and dialogue history. The context vector $c_j \in \mathbb{R}^{d_{hdd}}$ denotes the slot-specific information grabbed from the entire dialogue history. Finally, we obtain the context vectors $\mathbf{c} = [c_1, c_2, \cdots, c_J] \in \mathbb{R}^{d_{hdd} \times J}$ for all $J$ slots.

### 4.3 Slot Information Sharing

In the slot information sharing, there is a special matrix called the slot similarity matrix. This matrix controls the information flow among similar slots. We introduce two sharing methods according to their different calculation of the slot similarity matrix: fix combination sharing and the k-means sharing. We will compare the effectiveness of the two methods in Section 6.

#### 4.3.1 Fix Combination Method

We calculate the similarity between every two slots to construct switch matrix. We first compute the cosine similarity over the two slot names and then calculate the similarity over the slot types. Specifically, the slot types can be divided into several categories such as "date", "location". For example, if there are two slots "restaurant-area" and "restaurant-book day", then the similarity in the first part may be high since the two slot names share a common word "restaurant", while the similarity in the second part is quite low: slot "restaurant-area" has a value whose type is "location", and "restaurant-book day" has a value which belongs to "date". Next, the two calculated similarities $sname$ and $vtype$ will be integrated with a hyperparameter $\alpha \in [0, 1]$ and we can get a special matrix $sim \in \mathbb{R}^{J \times J}$ as a result.

$$sim = \alpha \cdot sname + (1 - \alpha) \cdot vtype \quad (4)$$

Here, the integration ratio $\alpha$ actually controls the final similarity of the slots. In Table 2, we show that different choices of this ratio will impact the model's tracking performance.

After that, matrix $sim$ is transformed into the slot similarity matrix $M$ by the mask mechanism.

$$M_{ij} = \begin{cases} 1 & if \quad sim_{ij} \geq \beta \\ 0 & if \quad sim_{ij} < \beta \end{cases} \quad (5)$$

Here, hyperparameter $\beta$ acts as a threshold to decide whether the two slots are similar enough to trigger the sharing switch and open the information path between them.

#### 4.3.2 K-means Sharing Method

Since the fix combination method needs manual efforts to search for the best hyperparameter, we propose another method, K-means Sharing Method, which requires no hyperparameter tuning and can achieve an averagely good performance. In this sharing method, we also compute the slot name similarity $sname_{ij}$ and the value type similarity $vtype_{ij}$ between slot $s_i$ and $s_j$ as the way in the fix combination one. Then we put vectors $(sname_{ij}, vtype_{ij})$ onto flat space and divide these vectors into two groups by the k-means clustering algorithm. One group stands for the slot $s_i$ and $s_j$ are similar enough, while the other one not similar. The element in $M_{ij}$ is 1 if they are in similar group, 0 if they are in unsimilar group.

After getting the slot similarity matrix whose value is either 1 or 0, we do the matrix multiplication between the context vectors $\mathbf{c} = [c_1, c_2, \cdots, c_J] \in \mathbb{R}^{d_{hdd} \times J}$ and the slot similarity matrix $M \in \mathbb{R}^{J \times J}$. Then we get the integrated vectors $\mathbf{int} = [int_1, int_2, \cdots, int_J] \in \mathbb{R}^{d_{hdd} \times J}$. These new vectors keep more expressive information for every slot. Specifically, $int_j$ is calculated as following:

$$int_j = \sum_{i=1}^{J} c_i \cdot M_{ij}, M_{ij} \in \{0, 1\} \quad (6)$$

As shown in the above equation, $int_j$ is essentially the integrated result of all related context vectors $c_i$ in $c$ and the integration is guided by the slot similarity matrix $M$. The matrix $M$ actually plays the role of a switch which controls the information flow between slots and provides a selective integration. For example, this integration makes the data-insufficient slot "attraction-type" receive the information from its related and data-sufficient slot "attraction-name", and helps our model deduce the related value for data-insufficient slots.

### 4.4 Decoder

The value prediction process of our decoder can be divided into two steps: first, predicting whether the value of a certain slot is constrained by the user; and then extracting the value if the constraint is mentioned in the dialogue.

In the first step, a three-way classifier called slot gate is used and it can map a vector taken from the encoded hidden states $H_t$ to a probability distribution over "ptr", "none", and "dontcare" labels. Once the slot gate predicts "ptr", the decoder will fill the slots with the values extracted from the dialogues. Otherwise, it just fills the slots with "not-mentioned" or "does not care".

In the second step, another GRU is utilized as the decoder. During the decoding step of the $j$-th slot, given a sequence of word embeddings $[w_{j_1}, w_{j_2}, \cdots, w_{j_{|N|}}]$, the GRU transforms them into decoded hidden states $[h^{dec}_{j_1}, h^{dec}_{j_2}, \cdots, h^{dec}_{j_{|N|}}]$ with the slot's integrated vector $int_j$:

$$z^j_k = \sigma(U_{z1} w_{j_k} + U_{z2} h^{dec}_{j_{k-1}}) \quad (7)$$

$$r^j_k = \sigma(U_{r1} w_{j_k} + U_{r2} h^{dec}_{j_{k-1}}) \quad (8)$$

$$\tilde{h^j_k} = \tanh(U_1 w_{j_k} + U_2(r^j_k \circ h^{dec}_{j_{k-1}})) \quad (9)$$

$$h^{dec}_{j_k} = (1 - z^j_k) \circ h^{dec}_{j_{k-1}} + z^j_k \circ \tilde{h^j_k} \quad (10)$$

Here, $|N|$ is the length of the slot sequence and $int_j$ is the initial hidden state input $h^{dec}_{j_0}$. The integrated vector $int_j$ makes the decoded hidden states contain more information about the dialogue history. So they are more sensitive about whether the value of slot $j$ is mentioned in the dialogue and where it locates. With the decoded hidden state $h^{dec}_{j_k}$, the generator computes $P^{gen}_{jk}$, the probability of the value generated from the vocabulary list $E \in \mathbb{R}^{|V| \times d_{hdd}}$ and $P^{copy}_{jk}$, the one copied from the interaction history. $|V|$ is the vocabulary size and $d_{hdd}$ is the dimension of the hidden state. In the end, we sum the probability $P^{gen}_{jk}$ and $P^{copy}_{jk}$ to yield the final prediction $P_{jk}$:

$$P^{gen}_{jk} = Softmax(E \cdot (h^{dec}_{jk})^\top) \quad (11)$$

$$P^{copy}_{jk} = Softmax(H_t \cdot (h^{dec}_{jk})^\top) \quad (12)$$

$$P_{jk} = g_{jk} \times P^{gen}_{jk} + (1 - g_{jk}) \times P^{copy}_{jk} \quad (13)$$

$$g_{jk} = Sigmoid(W_g \cdot [h^{dec}_{jk}; w_{jk}; P^{copy}_{jk} \cdot H_t]) \quad (14)$$

Here, $g_{jk}$ is a scalar which controls the model behaviour. It determines whether to generate values from the vocabulary list or copy words from the historical context.

## 5 Experiments

In this section, we first introduce the dataset and the evaluation metrics. We then describe our model's implementation details. Finally, we show our baseline models.

### 5.1 Datasets and Metrics

MultiWOZ (Budzianowski et al., 2018) is a fully-labelled collection of human-human written conversations spanning over multiple domains and topics. There are 7,032 multi-domain dialogues consisting of 2-5 domains in MultiWOZ. Because these dialogues have multiple tasks, so the long dialogue history makes state tracking more difficult. Since there are no dialogues from hospital and police domains in validation and testing sets of MultiWOZ, we follow TRADE (Wu et al., 2019) and use five out of the seven domains to train, valid and test, including restaurant, hotel, attraction, taxi and train. These domains involve 30 slots.

We also test our model on a subset of MultiWOZ which only contains the dialogues from the restaurant domain to verify whether our model still works for single-task dialogues.

We evaluate all the models using two metrics, slot accuracy and joint goal accuracy, similar to (Nouri and Hosseini-Asl, 2018):

- Slot accuracy. We use slot accuracy to check whether each single slot in the ground truth dialogue states is correct. The metric only focuses on if the slot requested is correct or not.

- Joint goal accuracy. The joint goal accuracy is used to evaluate whether the user's goal in each turn is captured. Only when every slot in the ground-truth dialogue state is considered and has correct value, can we consider the joint goal is achieved. It is the most important metric in the dialogue state tracking task.

### 5.2 Implementation Details

We use the concatenation embedding of GloVe embedding (Pennington et al., 2014) and the character-wise embedding (Hashimoto et al., 2017) in the

| Model | MultiWOZ | | MultiWOZ(res) | |
|---|---|---|---|---|
| | Joint | Slot | Joint | Slot |
| MDBT | 15.57 | 89.53 | 17.98 | 54.99 |
| GLAD | 35.57 | 95.44 | 53.23 | **96.54** |
| GCE | 36.27 | **98.42** | 60.93 | 95.85 |
| SpanPtr | 30.28 | 93.85 | 49.12 | 87.89 |
| TRADE | 48.62 | 96.92 | 65.35 | 93.28 |
| SAS | **51.03** | 97.20 | **67.34** | 93.83 |

Table 1: Performances of various models on MultiWOZ dataset and MultiWOZ (restaurant) dataset.

| Model | Joint | Slot |
|---|---|---|
| SAS-att-shr | 55.52 | 92.66 |
| SAS-shr | 60.68 | 89.53 |
| SAS(RT shr-0.7, 0.8) | 60.59 | 96.92 |
| SAS(RT shr-0.8, 0.7) | 60.78 | 96.94 |
| SAS(RT shr-0.8, 0.8) | 61.04 | **97.02** |
| SAS(RT shr-0.8, 0.9) | 60.54 | 96.91 |
| SAS(RT shr-0.9, 0.8) | **61.47** | 97.00 |
| SAS(KM shr) | 60.92 | 96.96 |
| SAS(HM shr) | 60.28 | 96.89 |

Table 2: Results evaluated on the MultiWOZ(except hotel) dataset. "RT shr" means the fix combination sharing method, "KM shr" is the k-means sharing method, and "HM shr" is the human evaluated sharing method. The two numbers after "-" represents the integration ratio $\alpha$ and the threshold $\beta$ respectively.

experiment. The model is trained with ADAM optimizer (Kingma and Ba, 2014) and a batch size of 32. Both the encoder and the decoder use 400 hidden dimensions. The learning rate is initially set to 0.001, but once the joint goal accuracy does not rise with the training, the network will automatically decrease its learning rate to improve the performance. We apply dropout with 0.2 dropout rate for regularization (Srivastava et al., 2014). Besides that, a word dropout technique is also utilized in the way proposed by (Bowman et al., 2015) which simulates the out-of-vocabulary setting. Our k-means clustering algorithm is implemented with the sklearn module, and we set all the hyperparameter in k-means algorithm as default.

### 5.3 Baseline Methods

We compare SAS with several previous methods: MDBT, GLAD, GCE, SpanPtr and TRADE. Based on the classical NBT model, MDBT (Ramadan et al., 2018) extended the task into multiple domains. MDBT makes full use of the semantic similarities between the dialogue and the slot ontology to track the domain and the value of the slot jointly. GLAD relies on global modules to learn the general information and local modules to catch the slot-specific information (Zhong et al., 2018) from the dialogues. GCE efficiently improves and simplifies GLAD, while keeping the excellent performance of GLAD (Nouri and Hosseini-Asl, 2018). SpanPtr first introduces the pointer network (Vinyals et al., 2015) into the dialogue state tracking task to extract unknown slot values (Xu and Hu, 2018). And in that paper, they also apply an effective dropout technique for training. TRADE directly generates slot values from the dialogues by using the copy mechanism and gets rid of the predefined value list (Wu et al., 2019). It achieves the previous state-of-the-art performance.

We use the fix combination version of SAS in Table 1 with the integration ratio $\alpha$ of 0.8 and the threshold $\beta$ of 0.8. That's the best hyperparameters we find for MultiWOZ.

## 6 Results

In this section, we first show the result of our model on MultiWoZ dataset, then on MultiWoZ(restaurant) and MultiWOZ (except hotel) dataset. After conducting the ablation experiment, we also display the improvement the slot attention and slot information sharing brings.

Our model achieves the best performance in the most important metric, joint goal accuracy. Our model outperformed the previous state-of-the-art model, TRADE by 2.41% absolute score on joint goal accuracy. We only observe slight increase of slot accuracy compared to TRADE. We suspect it is because TRADE was already achieving nearly 97% accuracy, which is close to the up-bound of the slot accuracy in this task. After carefully checking the error cases, we found these errors mainly come from the difficulty of generating name phrases.

To test SAS's ability on single domain dialogue tasks, we also evaluate our model on the a subset of MultiWOZ which contains only the restaurant search task. As displayed in Table 1, SAS achieved 1.99% improvement over TRADE on the joint goal accuracy as well, suggesting SAS's good performance generalize to single domain task.

Table 2 also shows how different choices of the hyperparameters influence the final results. On MultiWOZ, the integration ratio of 0.8 and the threshold of 0.8 are the best hyperparamters. But as

illustrated in Table 2, the best integration ratio is no longer 0.8 on MultiWOZ (except hotel). The best values of the integration ratio and the threshold will vary with the ontology.

We also perform ablation study to quantify different modules' contribution. We observe in Table 3 that adding the slot attention improves the state tracking results by 1.37% on MultiWOZ. Such improvement suggests having slot attention that focuses on the key information of the history is useful. And the slot information sharing further enhances the performance by 1.04%. The reason behind this may be that the information sharing of the related slots makes the data-insufficient slot receive more information. This handles the rare or unknown slot-value problems to some extent. As illustrated in Table 3, a model with the fix combination sharing method performs better than the k-means sharing method. But the fix combination method has an obvious shortcoming. It is difficult to generalize to new ontology. We need search the hyperparameters for every new ontology and these efforts are usually costly and time-consuming. Results in Table 2 and Table 3 indicate that the k-means algorithm provides a more robust model with respect to different parameters.

| | MultiWOZ | | MultiWOZ(res) | |
|---|---|---|---|---|
| **Model** | **Joint** | **Slot** | **Joint** | **Slot** |
| SAS-att-shr | 48.62 | 96.92 | 65.35 | 93.28 |
| SAS-shr | 49.99 | 97.10 | 66.89 | 93.62 |
| SAS(RT shr) | **51.03** | **97.20** | **67.34** | **93.83** |
| SAS(KM shr) | 50.46 | 97.15 | 66.65 | 93.78 |
| SAS(HM shr) | 50.27 | 97.13 | 66.89 | 93.62 |

Table 3: Performances of the models with different components on MultiWOZ dataset and MultiWOZ (restaurant) dataset. RT shr, KM shr, HM shr indicate the model is using the fix combination sharing method, k-means sharing method, and the human evaluated sharing method in the slot information sharing respectively.

To investigate whether the slot similarity matrices used by the two sharing methods really reflect the similarity among slots, we also compare them with a human constructed similarity matrix. We invite three volunteers to carefully rate (1 or 0) the relationship between every two slots and obtain the slot similarity matrix used in the human evaluated method. As shown in Table 2 and Table 3, the performance of the k-means sharing method is close to the one the human constructed method. This indicates human knowledge cannot further improve this task. Besides that, we also notice that the fix combination model usually outperforms the human constructed method, demonstrating that the fix combination model can automatically discover some hidden relationship among all slots that human cannot capture.

## 7 Error Analysis

To better understand why our model improves the performance, we investigated some dialogue examples and shown them in Table 4.

In the first dialogue, by asking "Could you also find me a hotel with a moderate price that offers internet?", the user has briefly informed the agent that he/she is looking for a hotel "with internet". The previous model missed the "hotel-internet" in the tracked slots. Because the model is mislead by the long interaction history. Our model learns to focus on important information using the slot attention to track the correct internet slot.

In the second dialogue, although the previous model manages to capture the value "21:30". It still confused "arriveby" with "leaveat". While SAS can distinguish them. We suspect this is because our model can learn the differences between these slots by training on isolated key features per slot without seeing any irrelevant information.

In the third example, the user agrees to visit an attraction named "Christ's College" from many college-type choices the agent suggests. Previous model fetches a wrong message and fills the slot "attraction-name" with "Clare College". In contrast, SAS captures the correct attraction name and also deduces that the attraction type is college. Similar to the first dialogue, the slot attention helps model gain more clean information to detect slot values more accurately. And by sharing the information fetched from slot "attraction-name" with the slot "attraction-type", our model is more sensitive with the value "college".

We also investigate the limitation of our model by analyzing the state tracking errors. We noticed two types of errors. First, SAS can not effectively identify value "dontcare" for most slots. For example, when the agent asks the user about his/her requirement on the hotel rating, though he/she answers "that is not really important for me", the model fails to fill "dontcare" into the slot "hotel-star". We believe this is due to the fact that the meaning of "dontcare" has plenty of expressions, it is much harder for the model to learn the semantic

| No | Model | Context |
|---|---|---|
| 1 | | I am looking for a train that leaves on saturday and arrives by 10:30. // Where are you traveling to and from? // · · · // Yes, that train sounds good. Please book it for me. Could you also find me a hotel with a moderate price that offers internet? // · · · // The north part of town please, preferably in a guesthouse. |
| | True | 'hotel-area-north', 'train-day-saturday', **'hotel-internet-yes'**, · · ·, 'hotel-pricerange-moderate', 'hotel-type-guest house' |
| | TRADE | 'train-arriveby-10:30', 'train-day-saturday', 'train-departure-birmingham new street', 'train-destination-cambridge', 'hotel-pricerange-moderate', 'hotel-type-guest house' |
| | SAS | 'train-destination-cambridge', 'train-departure-birmingham new street', · · · , **'hotel-internet-yes'**, 'train-arriveby-10:30' |
| 2 | | I am looking for a Chinese restaurant in the centre of town. // · · · // All Saints Church is famous for its architecture. It's located on Jesus Lane, cb58bs. They can be reached at 01223452587. Is there anything else I can find for you? // Yes. I need a taxi to take me from the church to the restaurant at 21:30. |
| | True | 'restaurant-food-chinese', 'attraction-area-centre', · · ·, 'taxi-departure-all saints church', 'restaurant-area-centre', **'taxi-leaveat-21:30'** |
| | TRADE | 'restaurant-food-chinese', 'attraction-area-centre', · · ·, **'taxi-arriveby-21:30'**, 'taxi-departure-all saints church', 'restaurant-area-centre', 'restaurant-pricerange-dontcare', **'taxi-leaveat-21:30'** |
| | SAS | 'taxi-destination-all saints church', 'restaurant-pricerange-dontcare', 'attraction-area-centre', **'taxi-leaveat-21:30'**, 'restaurant-food-chinese', 'taxi-departure-all saints church', 'attraction-name-all saints church', 'restaurant-area-centre' |
| 3 | | I would like to get some information about colleges to visit? // There is Christs College, Churchill College, Clare College , Clare Hall, Corpus Christi, Downing College, Emmanuel College, and Huges Hall. Would you like me to list more? // · · · // Tr6359 leaves at 13:40 and arrives 16:23, will this 1 work for you ? // Yes i need 6 tickets. |
| | True | **'attraction-type-college'**, 'train-departure-birmingham new street', · · ·, **'attraction-name-christ s college**, 'train-book people-6', 'train-day-friday', 'train-arriveby-16:30' |
| | TRADE | **'attraction-name-clare college'**, 'train-departure-birmingham new street', 'train-destination-cambridge', 'train-book people-6', 'train-day-friday', 'train-arriveby-16:30' |
| | SAS | 'train-destination-cambridge', 'train-departure-birmingham new street', **'attraction-name-christ s college'**, 'train-book people-6', · · · , **'attraction-type-college'** |

Table 4: Example dialogue state outputs from TRADE and SAS. "True" stands for ground truth dialogue states, "TRADE" and "SAS" are the generation results from TRADE and SAS respectively.

of "dontcare" than other slots. Besides that, we also notice that the tracking errors of departure or destination location are still common. The reason may be that location name words are usually rich in variations and have few grammatical feature.

## 8 Conclusions and Future Work

We present SAS, an effective DST model which successfully extracts the key feature from the original information excessive dialogue. The slot attention of SAS enables it to isolate the key information for each slot, while the slot information sharing enhances the expressiveness of the infor-

mation passed to each slot by integrating the information from similar slots. The sharing allows SAS to generalize on rare slot-value pairs with few training data. Our model reaches the state-of-the-art performance compared with previous models.

We believe that SAS provides promising potential extensions, such as adapting our model on other tasks where are troubled by excessive information. Besides that, we also notice that it is hard for SAS to correctly extract names of hotel or attraction which have rich variations. Designing a new model to address these problems may be our future work.

## Acknowledgement

This research is funded by the Science and Technology Commission of Shanghai Municipality (19511120200 & 18511105502) and by Xiaoi Research. The computation is performed in ECNU Multifunctional Platform for Innovation (001).

## References

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. Version 4.

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*, pages 5016–5026.

Danqi Chen. 2018. *Neural reading comprehension and beyond*. Ph.D. thesis, Stanford University.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint*, arXiv:1704.00051. Version 2.

Tongfei Chen, Chetan Naik, Hua He, Pushpendre Rastogi, and Lambert Mathias. 2019. Improving long distance slot carryover in spoken dialogue systems. *arXiv preprint*, arXiv:1906.01149. Version 1.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint*, arXiv:1412.3555. Version 1.

Franck Dernoncourt, Ji Young Lee, Trung H Bui, and Hung H Bui. 2017. Robust dialog state tracking for large ontologies. In *Dialogues with Social Robots*, pages 475–485. Springer.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint*, arXiv:1412.6980. Version 9.

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.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint*, arXiv:1812.00899. Version 1.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 305–314.

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

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)*, pages 561–568. IEEE.

Sanuj Sharma, Prafulla Kumar Choubey, and Ruihong Huang. 2019. Improving dialogue state tracking by discerning the relevant context. 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 576–581.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

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.

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*, volume 1, pages 438–449.

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.

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. Version 1.

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.

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.

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