Dialog State Tracking with Incorporation of Target Values in Attention Models

Takami Yoshida, Kenji Iwata, Yuka Kobayashi, and Hiroshi Fujimura

Abstract This paper proposes a fully data-driven approach to dialog state tracking (DST) that can handle new slot values not seen during training. The approach is based on a long short-term memory recurrent neural network with an attention mechanism. Unlike with conventional attention mechanisms, we use encoded user utterances and a hypothesis for the slot value (the target value) to calculate attention weights. In addition, while conventional attention mechanisms focus on words that correspond to trained values, the proposed attention mechanism focuses on words that correspond to a target value. Therefore, the DST model can detect unseen values by adding values to the hypothesis as target values. The proposed approach is evaluated using the second and the third Dialog State Tracking Challenge datasets. Evaluation results show that the proposed method improves 10.3 points on unseen slot values.

1 Introduction

In task-oriented spoken dialog systems, dialog state tracking (DST) is used to update a dialog state (i.e., the state of the user's goal). DST is an essential function for a dialog system because the state directly affects the system's response to the user. A dialog state is defined as "a data structure that summarizes the dialog history up to the time when the next system action is chosen" [1]. In practice, a dialog state is a probability distribution over a set of slots and their values as defined in a domain ontology. In the example shown in Table 1, "area, food, pricerange" are slots and "north, japanese,

T. Yoshida (✉) · K. Iwata · Y. Kobayashi · H. Fujimura
Media AI Laboratory, Corporate Research & Development Center, Toshiba Corporation, Kawasaki, Japan
e-mail: takami.yoshida@toshiba.co.jp

K. Iwata
e-mail: kenji4.iwata@toshiba.co.jp

Y. Kobayashi
e-mail: yuka3.kobayashi@toshiba.co.jp

H. Fujimura
e-mail: hiroshi4.fujimura@toshiba.co.jp

L. F. D'Haro et al. (eds.), *Conversational Dialogue Systems for the Next Decade*, Lecture Notes in Electrical Engineering 704, https://doi.org/10.1007/978-981-15-8395-7_9

Table 1 Examples of dialogs and dialog states

System response	User utterance	Dialog state
Hello. How may I help you? (*welcomemsg*())	Japanese restaurant	area=*none*[a], food=japanese, pricerange=none
What area do you prefer? (*request(area)*)	North part of town	area=north, food=japanese, pricerange=none
What price range do you prefer ? (*request(pricerange)*)	Any	area=north, food=japanese pricerange=*dontcare*[b]
How about XXX restaurant. (*offer(XXX)*)	Thank you	area=north, food=japanese pricerange=dontcare

[a] *none* means no value is specified.
[b] *dontcare* means user has no preference.

dontcare, none" are slot values. In practical applications, slot values may be changed during the operation of a dialog system. For example, in the restaurant information domain, the domain ontology changes when new restaurants are added. Therefore, DST should be able to handle a dynamic ontology and unseen slot values.

Traditional DST approaches use handcrafted rules [2, 3] because rule-based approaches are simple and intuitive. However, crafting rules is costly and applying them to a new domain is difficult. Recent DST approaches have been based on deep learning models such as recurrent neural networks (RNNs) [4–6], which need to be trained for predefined slots and values using the domain ontology. However, to deal with new or unseen slot values, training data must be prepared and a new model must be trained.

To overcome this drawback, DST models have been proposed that can handle unseen slot values without retraining [7–9]. The RNN models in [7] use input features after delexicalization. Delexicalization replaces the words relevant to slots and their values with generic symbols. However, delexicalization requires handcrafted rules that compare input words with a large list of synonyms for slots and their values. Another approach uses spoken language understanding (SLU) based on concept tagger architecture [8]. This approach utilizes slot names or slot descriptions to detect unseen values without model retraining. A neural belief tracker [9] estimates a dialog state by comparing the representations of user utterance, system response, and slot values. Although these methods generalize a DST model to unseen slot values, it comes at the cost of crafting the rules, the list of synonyms, slot description, or semantic dictionary.

A problem that is not adequately addressed in the literature is how to deal with unseen slot values without any handcrafted rules. Pointer network-based DST approaches [10, 11] can detect unseen values by utilizing context information. DST models use a pointer mechanism to extract words that are relevant to values. Although the model [10] showed comparatively good performance on the second Dialog State Tracking Challenge (DSTC2) dataset, the accuracies for unseen values were low.

Another DST model [11] showed better accuracies for unseen values in the third Dialog State Tracking Challenge (DSTC3) dataset. However, the results show the tradeoff between the accuracies for seen values versus those for unseen values. BERT-DST [12] extracts span (start position and end position) of the specified slot value from the user utterance and the system response. BERT-DST showed high accuracies for tracking unseen slot values. However, its effectiveness was evaluated using a restaurant name slot and a movie name slot; therefore, its effectiveness with other slots is unknown.

This paper proposes a new attention mechanism for a fully data-driven DST approach that can handle unseen slot values without handcrafted rules and model retraining. This approach is based on the pointer-based DST [11]. Unlike with conventional methods, we use encoded user utterances and a hypothesis for the slot values (the target values) to calculate attention. This enables the DST model to handle an unseen value by directly incorporating it into the attention weights. Attention weights are used to calculate context vectors, which are the weighted sums of word vectors. By comparing the context vectors and word vectors of slot values, the model estimates the dialog state. We evaluate the DST performance based on the proposed approach using the DSTC2 and DSTC3 datasets.

The remainder of this paper is organized as follows: Sect. 2 presents the proposed approach, Sect. 3 shows the experimental results and discusses their meaning and importance and Sect. 4 concludes the paper.

2 Dialog State Tracker

Our proposal is an extension of the DST model in [11], but differs from that approach in that target values are used to calculate attention. The new attention mechanism enables the model to focus on words that are relevant to the target values even if the target values were unseen in training.

Figure 1 illustrates an overview of our DST model. The model consists of encoding and decoding layers. The encoding layer extracts one score from system actions ($\boldsymbol{s}^{\mathrm{s}}$) and another score from user utterances ($\boldsymbol{s}^{\mathrm{u}}$) separately. These two scores are integrated with the previous dialog state ($\boldsymbol{s}^{\mathrm{p}}$) using weight parameters ($\boldsymbol{\beta} = [\beta^{\mathrm{s}}, \beta^{\mathrm{u}}, \beta^{\mathrm{p}}]$) in the decoding layer. The weighted sum ($\boldsymbol{y}$) is regarded as a probability distribution over the slot values after applying the softmax function.

We will describe DST models that use the conventional attention mechanism and the new target value attention mechanism in Sects. 2.1 and 2.2, respectively.

In the sections that follow, we explain a process for a particular slot that includes K values $(v_1, \ldots, v_K)$, in which the kth value consists of M_k words. We use n and N for the index of a word in a user utterance and the number of words in the user utterance, respectively.

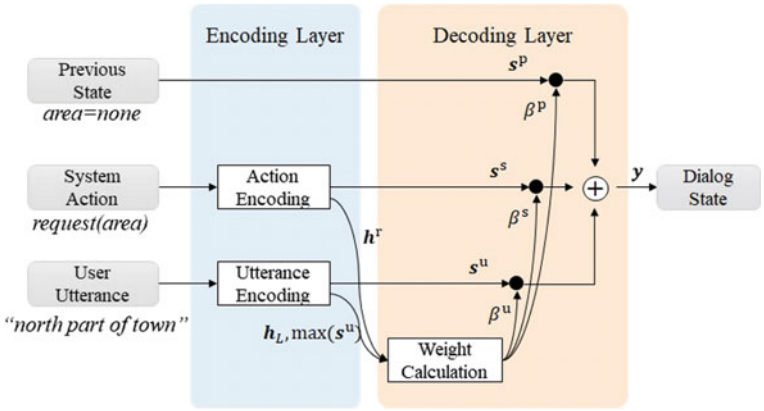


Fig. 1 Schematic diagram of our dialog state tracker

2.1 DST Model with Conventional Attention

This section describes a DST model with a conventional attention mechanism based on the model proposed in [11].

2.1.1 Encoding Layer

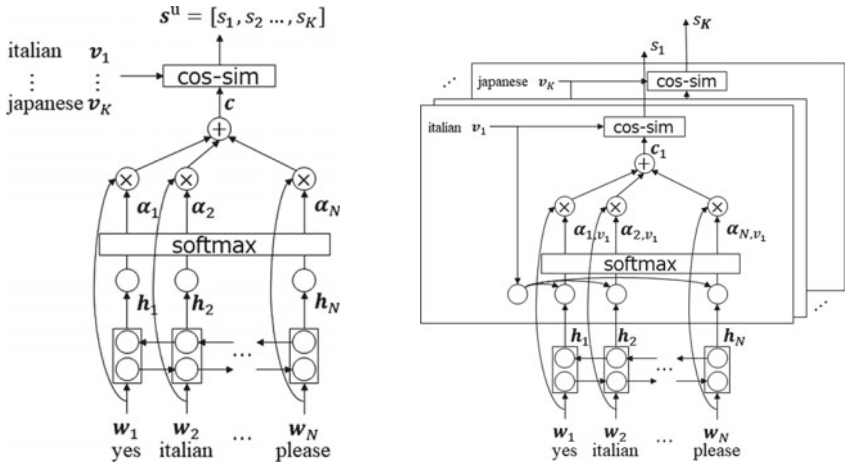
The utterance-encoding and the action-encoding modules calculate two kinds of features. One is possibility scores ($\boldsymbol{s}^{\mathrm{u}}, \boldsymbol{s}^{\mathrm{s}}$) that represent whether a slot value is the user's goal. The other is feature vectors ($\boldsymbol{h}^{\mathrm{r}}, \boldsymbol{h}_L$) for calculating weight parameters in the decoding layer.

Action Encoding

Action encoding extracts two kinds of features from a system action. One is a feature vector used for calculating weight parameters. The other is a score vector that represents how the system refers to a slot value.

The system action is represented as three features: a system action tag ($\boldsymbol{r}^{\mathrm{act}}$), a target slot feature (r^{s}), and a target value feature (r^{v}). The system action tag is a one-hot vector whose dimension is the same as the number of action tags. The target slot feature and the target value feature are 1 if the previous system action includes the target slot and target value and are 0 otherwise. The three features are concatenated and encoded using a neural network as:

$$\boldsymbol{h}^{\mathrm{r}} = \mathrm{NN}_{\mathrm{sys}}\left(\boldsymbol{r}^{\mathrm{act}} \oplus r^{\mathrm{s}} \oplus r^{\mathrm{v}}\right), \tag{1}$$



(a) Conventional attention mechanism (b) Target value attention mechanism

Fig. 2 Utterance encoding using attention mechanisms

where $\boldsymbol{h}^{\mathrm{r}} \in \mathbb{R}^{d_{\mathrm{m}}}$ is an output vector, d_{m} is a model parameter, $\mathrm{NN}_{\mathrm{sys}}(\cdot)$ is a fully-connected neural network and $\oplus$ is the vector concatenation. The output is used for the weight calculation.

The score vector of the system response ($\boldsymbol{s}^{\mathrm{s}}$) is a $K + 2$ dimensional binary flag. If the system response includes the kth value, the kth component is 1 and is 0 otherwise. The last two components correspond to special values "none" and "dontcare".

Utterance Encoding

Figure 2(a) shows a block diagram of utterance encoding, in which the module receives a user utterance and encodes it using an attention mechanism.

The user utterance of N word vectors is encoded using a bidirectional LSTM as follows:

$$\boldsymbol{h}_n^{\mathrm{f}} = \mathrm{LSTM}_{\mathrm{fwd}}(\boldsymbol{h}_{n-1}^{\mathrm{f}}, \boldsymbol{w}_n), \tag{2}$$

$$\boldsymbol{h}_n^{\mathrm{b}} = \mathrm{LSTM}_{\mathrm{bwd}}(\boldsymbol{h}_{n+1}^{\mathrm{b}}, \boldsymbol{w}_n), \tag{3}$$

$$\boldsymbol{h}_n = \boldsymbol{h}_n^{\mathrm{f}} \oplus \boldsymbol{h}_n^{\mathrm{b}}, \tag{4}$$

where $\boldsymbol{w}_n \in \mathbb{R}^{d_{\mathrm{w}}}, n = 1, \ldots, N$ is a word vector whose dimension is d_{w}, $\boldsymbol{h}_n^{\mathrm{f}} \in \mathbb{R}^{d_{\mathrm{m}}/2}$, $\boldsymbol{h}_n^{\mathrm{b}} \in \mathbb{R}^{d_{\mathrm{m}}/2}$, and $\boldsymbol{h}_n \in \mathbb{R}^{d_{\mathrm{m}}}$ are hidden states, $\mathrm{LSTM}_{\mathrm{fwd}}(\cdot, \cdot)$ and $\mathrm{LSTM}_{\mathrm{bwd}}(\cdot, \cdot)$ are forward and backward LSTM RNNs. Next, attention weights (α_n) are calculated from the hidden states ($\boldsymbol{h}_n$) as follows:

$$z_n = \mathrm{NN}_{\mathrm{att}}(\boldsymbol{h}_n), \tag{5}$$

$$[\alpha_1, \ldots, \alpha_N] = \mathrm{softmax}([z_1, \ldots, z_N]), \tag{6}$$

where $\mathrm{NN}_{\mathrm{att}}(\cdot)$ is a fully-connected neural network.

Then, a context vector ($\boldsymbol{c} \in \mathbb{R}^{d_{\mathrm{w}}}$) of the user utterance is calculated as a weighted sum of the word vectors as follows:

$$\boldsymbol{c} = \sum_{n=1}^{N} \alpha_n \boldsymbol{w}_n. \tag{7}$$

The score of the user utterance is calculated using cosine similarity between the context vector ($\boldsymbol{c}$) and the word vector of the kth value ($\boldsymbol{h}^{v_k} \in \mathbb{R}^{d_{\mathrm{w}}}$) as follows:

$$s_k = \frac{\boldsymbol{c} \cdot \boldsymbol{h}^{v_k}}{\|\boldsymbol{c}\| \|\boldsymbol{h}^{v_k}\|}. \tag{8}$$

Note that to handle values consisting of multiple words such as "eastern european", we use the sum of the word vectors as $\boldsymbol{h}^{v_k}$, that is, $\boldsymbol{h}^{v_k} = \sum_{m=1}^{M_k} \boldsymbol{v}_{k,m}$, where $\boldsymbol{v}_{k,m} \in \mathbb{R}^{d_{\mathrm{w}}}$ is the m-th word vector of the k-th value.

To estimate the scores ($\tilde{\boldsymbol{s}} = [s_{\mathrm{none}}, s_{\mathrm{dc}}]$) for the special values "*none*" and "*dontcare*", we use a separate neural network $\mathrm{NN}_{\mathrm{val}}(\cdot)$ as follows:

$$\boldsymbol{x} = \boldsymbol{h}^{\mathrm{f}}_{N^{\mathrm{u}}} \oplus \boldsymbol{h}^{\mathrm{b}}_1 \oplus \boldsymbol{h}^{\mathrm{r}} \oplus \max(\boldsymbol{s}^{\mathrm{u}}), \tag{9}$$

$$\tilde{\boldsymbol{s}} = \mathrm{NN}_{\mathrm{val}}(\boldsymbol{x}), \tag{10}$$

where $\boldsymbol{x} \in \mathbb{R}^{2d_{\mathrm{m}}+1}$ is the concatenation of the last states of the forward and backward LSTM ($\boldsymbol{h}^{\mathrm{f}}_N, \boldsymbol{h}^{\mathrm{b}}_0$), the system action feature ($\boldsymbol{h}^{\mathrm{r}}$), and the maximum cosine similarity ($\max(\boldsymbol{s}^{\mathrm{u}})$). Finally, the scores s_k and $\tilde{\boldsymbol{s}}$ are concatenated as $\boldsymbol{s}^{\mathrm{u}} = [s_1, \ldots, s_K] \oplus \tilde{\boldsymbol{s}}$. Note that we omit this part from Fig. 2(a) for simplicity. The utterance encoding sends the concatenated score ($\boldsymbol{s}^{\mathrm{u}}$) and feature vector ($\boldsymbol{x}$) to the following processing operation.

Decoding Layer

The decoding layer integrates scores calculated from the user utterance ($\boldsymbol{s}^{\mathrm{u}}$), the score from the system response ($\boldsymbol{s}^{\mathrm{s}}$), and the dialog state of the previous turn ($\boldsymbol{s}^{\mathrm{p}}$) using weight parameters ($\beta = [\beta^{\mathrm{u}}, \beta^{\mathrm{s}}, \beta^{\mathrm{p}}]$) from a neural network $\mathrm{NN}_{\mathrm{weight}}(\cdot)$ as follows:

$$\boldsymbol{\beta} = \mathrm{NN}_{\mathrm{weight}}(\boldsymbol{x}), \tag{11}$$

$$\boldsymbol{y} = \beta^{\mathrm{u}} \boldsymbol{s}^{\mathrm{u}} + \beta^{\mathrm{s}} \boldsymbol{s}^{\mathrm{s}} + \beta^{\mathrm{p}} \boldsymbol{s}^{\mathrm{p}}, \tag{12}$$

$$\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{y}). \tag{13}$$

After applying the softmax function, the model outputs the probability distribution ($\boldsymbol{p} \in \mathbb{R}^{K+2}$) over K slot values, none and dontcare.

2.2 DST Model with Target Value Attention

Figure 2(b) shows a block diagram of utterance encoding based on our attention mechanism. This model calculates attention weights for each target value using the word vectors of the corresponding target value. The attention mechanism is designed to focus the decoder on words in the user's utterance that are relevant to the target value. If the utterance includes a word relevant to the target value, the attention weight for the word will be greater than that for the other words. As a result, the context vector ($\boldsymbol{c}_k$) is similar to the word vector of the target value. Therefore, by comparing the context vector and the target value, the system can detect unseen values.

Instead of Eqs. (5), (6), attention weights (α_{n,v_k}) are calculated using the word vector of the k-th value ($\boldsymbol{h}^{v_k}$) and the hidden states ($\boldsymbol{h}_n$) as follows:

$$z_{n,v_k} = \mathrm{NN}_{\mathrm{att}}(\boldsymbol{h}^{v_k} \oplus \boldsymbol{h}_n), \tag{14}$$

$$\left[\alpha_{1,v_k}, \ldots, \alpha_{n,v_k}\right] = \mathrm{softmax}(\left[z_{1,v_k}, \ldots, z_{n,v_k}\right]), \tag{15}$$

where $\mathrm{NN}_{\mathrm{att}}(\cdot)$ is a fully-connected neural network.

Then, a context vector ($\boldsymbol{c}_k \in \mathbb{R}^{d_w}$) of the user utterance is calculated as a weighted sum of the word vectors as follow:

$$\boldsymbol{c}_k = \sum_{n=1}^{N^{\mathrm{u}}} \alpha_{n,v_k} \boldsymbol{w}_n. \tag{16}$$

Note that the context vector is calculated for each slot value.

The score of the user utterance is calculated using cosine similarity between the context vector ($\boldsymbol{c}_k$) and the value vectors ($\boldsymbol{h}^{v_k}$):

$$s_k = \frac{\boldsymbol{c}_k \cdot \boldsymbol{h}^{v_k}}{\|\boldsymbol{c}_k\| \|\boldsymbol{h}^{v_k}\|}. \tag{17}$$

The remaining parts are the same as the ones described in Sect. 2.1.

2.2.1 Model Training

When training the model, we minimize a loss function that consists of two terms. One is the cross-entropy loss (L_{ce}) between the output probabilities ($\boldsymbol{p}$) and the ground truth label ($\boldsymbol{d}$). The other is the triplet loss [13] (L_{tri}) between the normalized word vector of the ground truth value ($\boldsymbol{h}^{\mathrm{v}_\kappa}$) and the context vectors ($\boldsymbol{c} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K]$).

Table 2 Ontology of the train and test datasets

	Slot	Values		Examples
		All	Unseen	
Train	Area	7	-	Centre, North, ...
	Food	93	-	Afghan, African, ...
	Pricerange	5	-	Cheap, moderate, ...
Test	Area	17	14	Girton, arbury, ...
	Food	30	10	Cafe food, ...
	Pricerange	6	1	Free, cheap, ...

$$L = L_{\text{ce}}(\boldsymbol{d}, \boldsymbol{p}) + L_{\text{tri}}(\boldsymbol{h}^{\text{v}_\kappa}, \boldsymbol{c}), \tag{18}$$

$$L_{\text{ce}}(\boldsymbol{d}, \boldsymbol{p}) = -\sum \boldsymbol{d} \log \boldsymbol{p}, \tag{19}$$

$$L_{\text{tri}}(\boldsymbol{h}^{\text{v}_\kappa}, \boldsymbol{c}) = \frac{1}{K}\left(\sum_{j \neq \kappa} \max\left\{0, \|\boldsymbol{h}^{v_\kappa} - \boldsymbol{c}_\kappa\| - \|\boldsymbol{h}^{v_\kappa} - \boldsymbol{c}_j\| + \varepsilon\right\}\right), \tag{20}$$

where ε and κ are a margin parameter and an index of the ground truth value, respectively. The triplet loss helps the model learn the context vector calculation that assigns a smaller distance to $(\boldsymbol{h}^{\text{v}_\kappa}, \boldsymbol{c}_\kappa)$ and bigger distance to $(\boldsymbol{h}^{\text{v}_\kappa}, \boldsymbol{c}_{j \neq \kappa})$. Note that we add the triplet loss when the corresponding user utterance includes a slot value.

We introduce the "*Sampling with Decay*" technique [18] to feed the previous state $(\boldsymbol{s}^{\text{p}})$. During model training, we randomly sample the previous state from the ground truth previous state $(\boldsymbol{d})$ with a probability of q or from the estimated state $(\boldsymbol{p})$ with a probability of $1 - q$. We define q with the decay function dependent on the index of training epochs (e) as $q = \frac{\mu}{\mu + \exp(e/\mu)}$ where μ is a parameter. As the training proceeds, the probability of (q) feeding ground truth decreases gradually [18].

3 Experiments

We evaluated our model using the DSTC2 and DSTC3 datasets [14, 15]. The datasets include human-computer dialogs. Users interacted with dialog systems to search for restaurants by specifying constraints. Among the slots included in the DSTC3 dataset, we used "area", "food", and "pricerange". We excluded the "childrenallowed," "type," "hasinternet," "hastv," and "near" slots because these slots are not included in the DSTC2 dataset. We also excluded the "name" slot because word vectors for several values were not obtained. A summary of the slots and slot values are shown in Table 2. In the DSTC3 test dataset, 36.0%, 17.3%, and 3.5% of the dataset refer to unseen values in the area, food, and pricerange slots, respectively.

3.1 Experimental Condition

The evaluation metric is the accuracy of a value estimation. The accuracy is calculated as the fraction of turns where the top dialog state hypothesis is correct [14]. The ground truth label is under "Scheme A", which defines the label as the most recently asserted value and "Schedule 2"[14].

We implemented a prototype DST system based on the proposed method using a chainer [16]. One-best ASR results were used as inputs to the encoding layer described in Sect. 2. Contractions were converted to their original forms (e.g.., i'm to i am). Then, each word was converted to a 300-dimensional word vector using GloVe [17]. We used the GloVe model available at the GloVe website.[1] During training, the parameters of the GloVe model were fixed.

As a baseline method, we implemented a DST method that does not use the target value for attention weight calculation that is explained in Sect. 2.1. DST based on the RNN with the proposed attention mechanism and the conventional attention mechanism are called as "*Prop*" and "*Comp*", respectively.

We also evaluated two conventional methods, *BERT-based DST* [12] and *pointer-based DST* [11]. We implemented BERT-based DST using the publicly available BERT-DST source codes.[2] We used default parameters in the source code except the slot value dropout ratio. We trained models using the slot value dropout ratio of $[0, 0.1, \ldots, 0.4]$ and selected the best one. Note that the accuracies of the BERT-based DST were calculated using only pointable samples. The accuracies of the pointer-based DST are the ones reported in [11].

For $\mathrm{LSTM}_{\mathrm{fwd}}$ and $\mathrm{LSTM}_{\mathrm{bwd}}$, we used 1-layer LSTMs with 32 nodes. For $\mathrm{NN}_{\mathrm{sys}}$, $\mathrm{NN}_{\mathrm{att}}$, $\mathrm{NN}_{\mathrm{val}}$, and $\mathrm{NN}_{\mathrm{weight}}$, we used 1-layer, 3-layer, 4-layer, and 4-layer fully connected NNs, respectively.

Hyper parameters are as follows: Adam optimizer using the chainer implementation; learning rate, 0.001; gradient clipping, 1.0; mini batch size, 32; sampling parameter μ, 12; and maximum epoch, 200. We also applied word dropout that randomly replaced the word vectors of user utterances with zero vectors. These hyper parameters were identical for the Comp and Prop models.

3.2 Results and Discussion

Table 3 shows DST accuracies on the DSTC2 dataset. The upper part shows the results reported in the DSTC2 [14] and the lower part shows the results of fully data-driven DST methods. In all slots, RNN with rules shows the best performance; from 0.2 to 0.7 point higher than Comp and Prop. Prop and Comp achieve almost the same performance as Focus baseline without using any handcrafted rules. The differences between Comp and Prop is less than 0.4 point. This is reasonable because Comp

[1] https://nlp.stanford.edu/projects/glove/.

[2] https://github.com/guanlinchao/bert-dst.

Table 3 Test accuracies on the DSTC2 test dataset

Tracker	Area	Food	Price
Focus baseline [14]	90.8	83.9	92.9
RNN with rules [7]	**92.4**	**85.6**	**93.0**
Pointer-based DST [11]	84.7	84.4	83.7
BERT-based DST	88.2	76.6	92.0
Comp	91.7	84.9	92.5
Prop	**91.9**	**85.1**	**92.8**

Table 4 Test accuracies on the DSTC3 test dataset

Tracker	All values			Unseen values		
	Area	Food	Price	Area	Food	Price
Focus baseline [14]	81.1	90.5	88.4	67.6	**88.1**	87.6
RNN with rules [7]	**88.5**	**91.0**	**93.2**	**85.3**	82.3	**92.3**
Pointer-based DST [11]	**80.6**	79.6	66.9	**71.5**	59.5	52.7
BERT-based DST [12]	61.3	79.2	90.4	25.8	61.9	49.4
Comp	75.2	**83.5**	91.4	55.7	66.2	52.7
Prop	76.1	**83.5**	**91.7**	57.8	**71.3**	**79.6**

can extract words relevant to seen values without using the target value attention mechanism.

Table 4 shows DST accuracies on the DSTC3 test dataset. This table reveals the gap between rule-based DST and fully-data driven DST models. Among fully-data driven models, Comp shows better performance than does BERT-based DST under all conditions. Prop improves accuracies further. Pointer-based DST shows high accuracies on the area slot, but the accuracies are lower on the other slot.

The accuracy of Prop on the area slot is lower than that on the food and pricerange slots. The lower score might be caused by values consisting of multiple words such as “new chesterton”, “kings hedges”. In the training dataset, the area slot includes values consisting of single words such as “north” and “south”. Therefore, the DST models learned to extract only single words from user utterances. We observed that BERT-based DST also suffers from such word length mismatches.

We perform ablation experiments on the DSTC3 test dataset to analyze the effectiveness of different components. The results are shown in Table 5. The accuracies

Table 5 Ablation study on the DSTC3 test dataset

Tracker	All values			Unseen values		
	Area	Food	Price	Area	Food	Price
without TVA[a] (Comp)	75.2	83.5	91.4	55.7	66.2	52.7
+ SVD[b]	**76.5**	**84.6**	91.4	**59.2**	65.9	52.7
with TVA	73.3	80.6	91.1	57.8	66.5	76.2
+ SVD	76.1	82.1	91.6	57.8	66.5	76.2
+ SVD + TL[c] (Prop)	76.1	83.5	**91.7**	57.8	**71.3**	**79.6**

[a] TVA represent target value attention
[b] SVD represent slot value dropout
[c] TL represent triplet loss

of "without TVA" and "with TVA" are almost the same. However, integration of the three components achieves the comparative or higher accuracies under most conditions. The effect of our method is more pronounced on unseen values than all values. On average over the three slots, the score of prop (69.6) is 10.3 points higher than that of Comp + SVD (59.3).

One of the drawbacks is that our method requires a word vector for the target value. This method cannot track values whose word vector is not available. This is why we exclude the name slot for the experiments. One promising approach is to use a part of a word as a unit.

Another drawback is that the proposed model tends to fail when two values include the same word. In the test dataset, "pub" is included in the food slot ("pub food") and the type slot ("pub"). When a user says "I'm looking for a pub food restaurant," the ground truth of the dialog state is "food = pub food, type = pub." On the other hand, if a user says "I'm looking for a pub," the ground truth of the dialog state is "food = none, type = pub." The DST model with target value attention tends to estimate such user utterances as "food = pub food."

4 Summary

This paper proposed a fully data-driven approach to DST based on a target value attention mechanism. Unlike conventional attention mechanisms, the proposed attention mechanism utilizes the hypothesis for slot values in order to focus on unseen values without model retraining. We used the DSTC2 and DSTC3 datasets to evaluate the DST model based on the proposed approach. For unseen values, the results showed that using the proposed attention mechanism led to a 10.3-point improvement over the conventional attention mechanism.

Future research will aim to improve the accuracy of the model for both seen and unseen values as well as extend the proposed approach to handle unseen slots.

Acknowledgements The authors would like to thank Professor Masami Akamine for his insightful comments and suggestions.

References

1. Williams JD, Raux A, Henderson M (2016) The dialog state tracking challenge series: a review. Dialogue Discourse 7(3):4–33
2. Sun K, Chen L, Zhu S, Yu K (2014) A generalized rule based tracker for dialogue state tracking. In: Spoken language technology workshop (SLT). IEEE, pp 330–335
3. Kadlec R, Vodolan M, Libovickỳ J, Macek J, Kleindienst J (2014) Knowledge-based dialog state tracking. In: Spoken language technology workshop (SLT). IEEE, pp 348–353
4. Yoshino K, Hiraoka T, Neubig G, Nakamura S (2016) Dialogue state tracking using long short term memory neural networks. In: Proceedings of seventh international workshop on spoken dialog systems (IWSDS), pp 1–8
5. Zilka L, Jurcicek F (2015) Incremental LSTM-based dialog state tracker. In: Workshop on automatic speech recognition and understanding (ASRU). IEEE, pp 757–762
6. Liu B, Lane I (2017) An end-to-end trainable neural network model with belief tracking for task-oriented dialog. Proc Interspeech 2017:2506–2510
7. Henderson M, Thomson B, Young S (2014) Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In: Spoken language technology workshop (SLT). IEEE, pp 360–365
8. Rastogi A, Hakkani-Tür D, Heck L (2017) Scalable multi-domain dialogue state tracking. In: Automatic speech recognition and understanding workshop (ASRU). IEEE, pp 561–568
9. Mrkšić N, Vulić I (2018) Fully statistical neural belief tracking. In: Proceedings of the 56th annual meeting of the association for computational linguistics, vol 2, pp 108–113
10. Xu P, Hu Q (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, vol 1, pp 1448–1457
11. Yoshida T, Iwata K, Fujimura H, Akamine M (2018) Dialog state tracking for unseen values using an extended attention mechanism. In: International workshop on spoken dialog system technology (IWSDS)
12. Chao G, Lane I (2019) BERT-DST: scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. In: Proceedings of interspeech, pp 1468–1472
13. Wang J, Song Y, Leung T, Rosenberg C, Wang J, Philbin J, Chen B, Wu Y (2014) Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1386–1393
14. Henderson M, Thomson B, Williams J (2014) The second dialog state tracking challenge. In 15th annual meeting of the special interest group on discourse and dialogue, vol 263
15. Henderson M, Thomson B, Williams JD (2014) The third dialog state tracking challenge. In: Spoken language technology workshop (SLT). IEEE, pp 324–329
16. Tokui S, Oono K, Hido S, Clayton J (2015) Chainer: a next-generation open source framework for deep learning. In: *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*
17. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Empirical methods in natural language processing (EMNLP), pp 1532–1543
18. Zhang W, Feng Y, Meng F, You D, Liu Q (2019) Bridging the gap between training and inference for neural machine translation. In: *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp 4334–4343