Dialog State Tracking for Unseen Values using an Extended Attention Mechanism

Takami Yoshida, Kenji Iwata, Hiroshi Fujimura and Masami Akamine

Abstract Recently, discriminative models using recurrent neural networks (RNNs) have shown good performance for dialog state tracking (DST). However, the models have difficulty in handling new dialog states unseen in model training. This paper proposes a fully data-driven approach to DST that can deal with unseen dialog states. The approach is based on an RNN with an attention mechanism. The model integrates two variants of RNNs: a decoder that detects an unseen value from a user's utterance using cosine similarity between word vectors of the user's utterance and that of the unseen value; and a sentinel mixture architecture that merges estimated dialog states of the previous turn and the current turn. We evaluated the proposed method using the second and the third dialog state tracking challenge (DSTC 2 and DSTC 3) datasets. Experimental results show that the proposed method achieved DST accuracy of 80.0% for all datasets and 61.2% for only unseen dataset without hand-crafted rules and re-training. For the unseen dataset, the use of the cosine similarity-based decoder leads to a 26.0-point improvement from conventional neural network-based DST. Moreover, the integration of the cosine similarity-based decoder and the sentinel mixture architecture leads to a further 2.1-point improvement.

1 Introduction

Dialog state tracking (DST) is used to update the state of a user's goal, or a dialog state in a task-oriented spoken dialog system. A dialog state is, in practice, a probability distribution of slots and their values that are provided in an ontology. The dialog state is estimated by using a dialog history including the user's utterances and the system's responses up to the present point in time as shown in Table 1. DST

Media AI Laboratory, Corporate Research & Development Center, Toshiba Corporation,
e-mail: {takami.yoshida, kenji4.iwata, hiroshi4.fujimura, masa.akamine}@toshiba.co.jp

Table 1 Example of dialog and dialog states

	System response (*tag*)	User's utterance	Dialog state		
			Area	Food	Price
1	Hello. How may I help you? (*welcomemsg()*)	Cheap restaurant in the south part of town.	south	*None**	cheap
2	What kind of food would you like? (*request(food)*)	Any.	south	dontcare	cheap
3	How about the XX restaurant. (*offer(name = XX)*)	What is the phone number?	south	dontcare	cheap
4	Phone number is 1234. (*inform(phone = 1234)*)	Thank you, good bye.	south	dontcare	cheap

$*$*None* means no value is specified.

is one of the essential functions of a dialog system because the state directly affects the system's decision on its action with respect to the user.

Many approaches to DST have been proposed. They can be classified into three categories: rule-based, generative model-based, and discriminative model-based approaches. The rule-based approach [15] is simple and intuitive. However, making rules is time-consuming and money-consuming, especially when the task is complicated. The generative model-based [13] and the discriminative model-based [3, 4, 5, 6, 9, 12] approaches do not need hand-crafted rules because they are statistical and data-driven. Compared to generative models, discriminative ones have shown better performance in the dialog state tracking challenges, DSTC 2 [1] and DSTC 3 [2].

Discriminative models estimate a probability distribution of pre-defined values. Recent discriminative model-based approaches have used neural network models including a feedforward deep neural network (DNN) [3], a recurrent neural network (RNN) [4, 5, 6], and a convolutional neural network (CNN) [12].

However, the discriminative model-based approach has a drawback. Discriminative models need to be trained for slots and values predefined in the ontology of the system. To deal with a new dialog state, we need to prepare training data and train a new model.

To overcome this drawback, two RNN-based DST methods have been proposed. Henderson et al. proposed an RNN-based method with n-gram type features of delexicalized slots and values [4, 5]. The RNN takes input features of n-grams from hypotheses of automatic speech recognition (ASR) and system actions after delexicalization of words relevant to slots and their values. The method showed high accuracy for unseen values in DSTC 3. However, this delexicalization requires a large list of synonyms for slots and their values. Since the list is hand-crafted, the approach is time-consuming.

Jang et al. proposed a DST method based on an RNN with an attention mechanism [7]. The attention mechanism calculates weight parameters called attention weights that represent the importance of each word. The method calculates a context vector, which is a weighted sum of word vectors using attention weights. The

RNN estimates probability of values using cosine similarity between the context vector and a word vector of each value. The method can detect an unseen value in training from a user's utterance. However, [7] does not describe how to integrate dialog states of the previous turn and the current turn because the task does not require such integration.

An interesting variation of an attention mechanism has been proposed for language modeling [10]. Language modeling involves estimating the next word in a sentence from the previous words. The paper [10] introduces a mixture model that combines an RNN-based language model and a pointer model. The former model estimates probability of the next word over vocabulary words as standard language models and the latter probability of the next word that does not appear in training but in the sentence. The combination is done using an attention mechanism called "*sentinel mixture architecture*."

This paper proposes a fully data-driven approach to DST that can track an unseen value without hand-crafted rules and re-training. The approach uses an RNN with an attention mechanism based on cosine similarity [7] to detect an unseen value from a user's utterance. We introduce a sentinel mixture architecture [10] to the RNN model for integrating dialog states of the previous turn and the current turn. We evaluated the proposed method using the DSTC 2 and DSTC 3 datasets.

This paper is organized as follows: Sect. 2 presents the background, Sect. 3 sets out the proposed approach, Sect. 4 details experimental results, and Sect. 5 concludes the paper.

2 Related Work

Jang et al. proposed DST based on an RNN model with an attention mechanism [7] for the fifth dialog state tracking challenge (DSTC 5) [8]. The main task is to estimate a dialog state for each sub-dialog segment consisted of several turns that show the same topic. The main difficulty is the sparsity of the training data: only 56% of values in the ontology is included in the training data [7]. Therefore, DST is required to estimate values unseen in training. To realize such DST, [7] proposed DST based on an RNN with an attention mechanism. The method first calculates an attention weight for each word of a user's utterance using a long short-term memory (LSTM). Then it calculates cosine similarity between a context vector, i.e., a weighted sum of word vectors using the attention weights, and a word vector of each value in the ontology. Since the method estimates a dialog state based on cosine similarities, the DST can estimate a dialog state unseen in training to the extent that the word vector of the unseen value is available. This approach is effective in handling unseen values.

However, in contrast with DSTC 5, our DST needs to estimate a dialog state for each turn as in DSTC 2 and DSTC 3, and consider the dialog state of the previous turn. The second turn in Table 1 provides an example. Although the user does not mention the area slot, the ground truth of the dialog state keeps "$area = south$,"

which is mentioned in the previous turn. This example shows that a dialog state is decided by not only the user's utterance of the current turn, but also by a dialog state of the previous turn. The DST in [7] can not handle such situation.

Merity et al. proposed an RNN with a sentinel mixture architecture for language modeling [10]. As mentioned in [10], recent RNN-based language models show high performance but struggle to predict unseen words. To accurately predict both seen and unseen words, [10] proposed an integration of probability for words from a pre-defined vocabulary and probability for words appeared in input sentences. The former is calculated using an RNN like conventional approaches: the RNN outputs the probability of the next word over vocabulary words given previous words in a sentence. The latter is calculated using an attention mechanism: the model regards an attention weight for each word not seen in training but in the input sentence as the probability of the corresponding word. Therefore, the method can predict a word unseen in training if the word is included in the input. The two probabilities are summed up with a weight parameter called a "*sentinel weight*," which is calculated using the attention mechanism. The sentinel weight makes it possible to accurately predict both seen and unseen words, and their approach achieved state-of-the-art performance.

However, unlike language modeling, DST should bring a value and its synonyms together. For example, both "a restaurant in the center of the town" and "a restaurant in the central part of the town" indicate "$area = center$." Therefore, their method should be modified for DST.

To deal with unseen values as well as seen values in DST, we propose to introduce a sentinel mixture architecture to a cosine similarity-based decoder. We integrate a dialog state of the previous turn and the current turn using the sentinel architecture [10]. To the best of our knowledge, there is no previous study on fully data-driven DST that can track an unseen value.

3 Proposed RNN with Attention Mechanism

Figure 1 shows a schematic block diagram of the proposed model. The model consists of an encoding layer and a decoding layer. The model also includes an attention mechanism and sentinel mixture architecture. The model calculates an attention weight for each word and two sentinel weights. The attention weights are used in the attention mechanism and the sentinel weights are used in the sentinel mixture architecture.

The model receives a system response and a user's utterance. The system response consists of a system action tag (e.g., *welcomemsg, confirm,* ...), a target slot (e.g., *area, food,* ...), and a target value (e.g., *north, south,* ...). Examples of the system response include $welcomemsg()$, $confirm(food = italian)$, $inform(area = north)$. The user's utterance is a word sequence (e.g., "yes italian restaurant please"). The model includes a recurrent connection that receives a raw

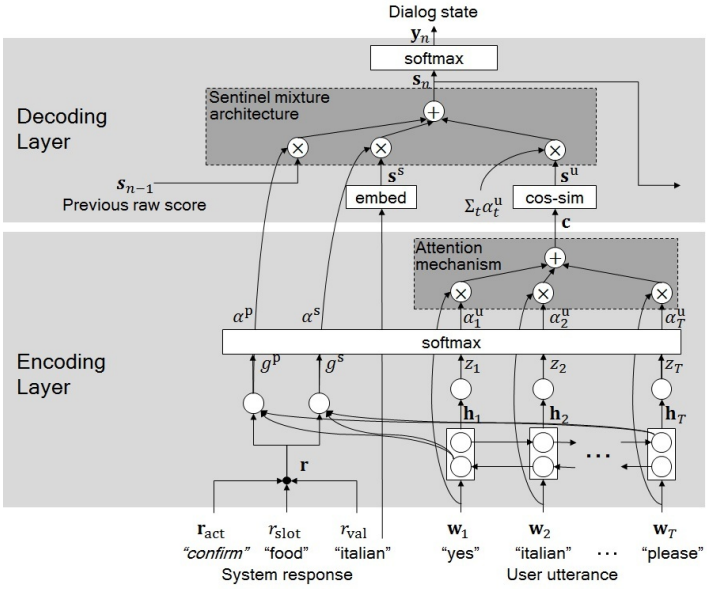

Fig. 1 Model architecture based on the proposed method

score. The raw score denotes a reliability that becomes the probability of each value after normalization using a softmax function.

We assume that the probability distribution can be factorized by that of each slot ($P(\text{area}, \text{food}, \text{price}) = P(\text{area})P(\text{food})P(\text{price})$). We describe how to estimate the probability for a slot in the following sections.

In the following sections, we assume that the DST focuses on the n-th turn, a slot has K values, the utterance consists of T words, and the system can take M actions.

3.1 Encoding Layer

The encoding layer calculates a context vector and sentinel weights from the system response and the user's utterance.

A system action tag, a target slot, and a target value are converted into an action tag vector, a target slot feature, and a target value feature, respectively. The action tag vector is a one-hot vector $\mathbf{r}_{\text{act}}$ whose dimension is M. The target slot feature r_{slot} is a binary value that represents whether the system response includes the slot ($r_{\text{slot}} = 1$) or not ($r_{\text{slot}} = 0$). The target value feature r_{val} is also a binary value that represents whether the system response includes any value ($r_{\text{val}} = 1$) or not ($r_{\text{val}} = 0$). For example, the features $(r_{\text{slot}}, r_{\text{val}})$ for *welcomemsg()*, *request(food)*, and *confirm(food=dontcare)* are $(0,0)$, $(1,0)$, and $(1,1)$, respectively.

The user's utterance is converted into word vectors ($\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T$). The word vectors are then converted into hidden state vectors ($\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T$) using a bi-directional LSTM as follows:

$$\begin{aligned}
\mathbf{h}_t^{\mathrm{f}} &= \mathrm{LSTM}_{\mathrm{fwd}}\left(\mathbf{h}_{t-1}^{\mathrm{f}}, \mathbf{w}_t\right) , \\
\mathbf{h}_t^{\mathrm{b}} &= \mathrm{LSTM}_{\mathrm{bwd}}\left(\mathbf{h}_{t+1}^{\mathrm{b}}, \mathbf{w}_t\right) , \\
\mathbf{h}_t &= \mathbf{h}_t^{\mathrm{f}} \oplus \mathbf{h}_t^{\mathrm{b}} ,
\end{aligned}$$

where $\mathrm{LSTM}_{\mathrm{fwd}}(\cdot,\cdot)$, $\mathrm{LSTM}_{\mathrm{bwd}}(\cdot,\cdot)$ are forward and backward LSTMs, $\mathbf{h}_t^{\mathrm{f}}, \mathbf{h}_t^{\mathrm{b}}$ are hidden states of forward and backward LSTMs, and $\oplus$ is vector concatenation. The hidden state vectors are used to calculate scalar values ($z_1, z_2, \ldots, z_T$) using a one-layer neural network ($\mathrm{NN}_{\mathrm{usr}}$) as follows:

$$z_t = \mathrm{NN}_{\mathrm{usr}}\left(\mathbf{h}_t\right) .$$

The scalar value z_t indicates the importance of the t-th word. A larger z_t means the t-th word is more important.

To calculate the sentinel weights, two sentinel gate parameters ($g^{\mathrm{p}}, g^{\mathrm{s}}$) are calculated from the system feature vector ($\mathbf{r} = \mathbf{r}_{\mathrm{act}} \oplus r_{\mathrm{slot}} \oplus r_{\mathrm{val}}$) and the hidden state ($\mathbf{h}_{\mathrm{L}} = \mathbf{h}_T^{\mathrm{f}} \oplus \mathbf{h}_1^{\mathrm{b}}$) as follows:

$$\begin{aligned}
g^{\mathrm{p}} &= \mathrm{NN}_{\mathrm{pre}}\left(\mathbf{h}_{\mathrm{L}} \oplus \mathbf{r}\right) , \\
g^{\mathrm{s}} &= \mathrm{NN}_{\mathrm{sys}}\left(\mathbf{h}_{\mathrm{L}} \oplus \mathbf{r}\right) ,
\end{aligned}$$

where $\mathrm{NN}_{\mathrm{pre}}, \mathrm{NN}_{\mathrm{sys}}$ are one-layer neural networks. The sentinel gate parameters indicate the importance of the raw score in the previous turn and that of the system response. A larger gate parameter means the corresponding feature vector is more important. Two sentinel weights ($\alpha^{\mathrm{p}}, \alpha^{\mathrm{s}}$) and attention weighs for the words ($\alpha_1^{\mathrm{u}}, \alpha_2^{\mathrm{u}}, \ldots, \alpha_T^{\mathrm{u}}$) are calculated from g^{p}, g^{s}, and $\mathbf{z} = [z_1, z_2, \ldots, z_T]$ as follows:

$$\alpha = \mathrm{softmax}\left(g^{\mathrm{p}} \oplus g^{\mathrm{s}} \oplus \mathbf{z}\right) , \tag{1}$$

where $\alpha = [\alpha^{\mathrm{p}}, \alpha^{\mathrm{s}}, \alpha_1^{\mathrm{u}}, \alpha_2^{\mathrm{u}}, \ldots, \alpha_T^{\mathrm{u}}]$.

Using α, the model focuses on the corresponding context in estimating dialog states. For example, in estimation of the area slot at the first turn of the dialog shown in Table 1, a dialog state is decided from the fifth word of the user's utterance, "south." The model focuses on the word by setting $\alpha_5^{\mathrm{u}} = 1$. At the second turn, the dialog state is decided from that of the previous turn. Thus, the model focuses on the raw score of the previous turn by setting $\alpha^{\mathrm{p}} = 1$. When the user refers to his or her goal by answering the system's question (e.g., the system asks "Let me confirm. You are looking for a venue in the Newnham area." and the user answers "yes"), the model focuses on the system's action by setting $\alpha^{\mathrm{s}} = 1$.

From the word vector sequence, the context vector $\mathbf{c}$ is calculated using the attention weights ($\alpha_1^{\mathrm{u}}, \alpha_2^{\mathrm{u}}, \ldots, \alpha_T^{\mathrm{u}}$) as follows:

$$\mathbf{c} = \sum_{t=1}^{T} \alpha_t^{\mathrm{u}} \mathbf{w}_t \ . \tag{2}$$

Equation (2) is described as the processing block "*Attention mechanism*" in Fig. 1. When the user's utterance includes a word related to a value, the corresponding attention weight becomes 1 and the context vector is the same as the word vector of the word. Therefore, by comparing the context vector and the word vector of the target value, the system can detect an unseen value in the user's utterance.

3.2 Decoding Layer

The decoding layer estimates a dialog state using the previous raw score, the system response, and the context vector with the sentinel weights. The decoding layer calculates two similarities: a similarity between the context vector and a word vector of each value, and a similarity between the system response and a value. These two similarities and the previous raw score are integrated with the sentinel weights.

From the system response, the decoding layer extracts a binary vector ($\mathbf{s}^{\mathrm{s}} = [s_1^{\mathrm{s}}, s_2^{\mathrm{s}}, \ldots, s_K^{\mathrm{s}}]$) whose k-th component indicates whether the k-th value is included ($s_k^{\mathrm{s}} = 1$) or not ($s_k^{\mathrm{s}} = 0$). This process is described as the processing block "*embed*" in Fig. 1.

Cosine similarity ($\mathbf{s}^{\mathrm{u}}$) between the context vector $\mathbf{c}$ and word vectors of each value ($\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K$) is calculated as follows:

$$s_k^{\mathrm{u}} = \frac{\mathbf{c} \cdot \mathbf{v}_k}{\|\mathbf{c}\| \|\mathbf{v}_k\|} \ , \tag{3}$$

$$\mathbf{s}^{\mathrm{u}} = [s_1^{\mathrm{u}}, s_2^{\mathrm{u}}, \ldots, s_K^{\mathrm{u}}] \ , \tag{4}$$

where $\cdot$ is dot product. If a value consists of more than two words (e.g., *chinese takeaway*), we used the average of the word vectors in Eq. (3). These Eqs. (3) and (4) are represented as the processing block "*cos-sim*" in Fig. 1.

The sentinel weights ($\alpha^{\mathrm{p}}, \alpha^{\mathrm{s}}, \alpha^{\mathrm{u}} = \sum_{t=1}^{T} \alpha_t^{\mathrm{u}}$) control integration of the raw score and the similarities ($\mathbf{s}_{n-1}, \mathbf{s}^{\mathrm{s}}, \mathbf{s}^{\mathrm{u}}$) as follows:

$$\mathbf{s}_n = \alpha^{\mathrm{p}} \mathbf{s}_{n-1} + \alpha^{\mathrm{s}} \mathbf{s}^{\mathrm{s}} + \alpha^{\mathrm{u}} \mathbf{s}^{\mathrm{u}} \ . \tag{5}$$

Equation (5) is described as the processing block "*Sentinel mixture architecture*" in Fig. 1.

A bias b is calculated as $b = \mathrm{NN}_{\mathrm{bias}}(\mathbf{h}_{\mathrm{L}} \oplus \mathbf{r})$ where $\mathrm{NN}_{\mathrm{bias}}$ is a one-layer neural network. The bias is used to estimate probability for None that means no value is specified. Note that we omit this bias calculation from Fig. 1 for simplicity. Finally, we calculate output probability $\mathbf{y}_n$ from the concatenation of the current raw score $\mathbf{s}_n$ and the bias b as follows:

$$\mathbf{y}_n = \mathrm{softmax}(\mathbf{s}_n \oplus b) \ . \tag{6}$$

Table 2 Values included in DSTC 2 and DSTC 3 datasets

	Training	DSTC 2 Test set		DSTC 3 Test set		
	Values	Seen	Unseen	Seen	Unseen	Examples of unseen
area	7	7	0	3	14	girton, arbury, ...
food	92	92	0	20	10	american, cafe food, ...
price range	5	5	0	6	1	free

4 Experiments

We evaluated the proposed method through two tasks, "Task 1" and "Task 2." DST performance for seen values was evaluated in Task 1 using the DSTC 2 dataset. In Task 2, DST performance for both seen and unseen values was evaluated using the DSTC 3 dataset. These two datasets were collected in advance, and the system performs DST for the datasets.

4.1 Datasets

The datasets include human–computer dialogs. Users interacted with dialog systems to search for restaurants by specifying constraints. A ground truth of a dialog state was labeled for all turns as the most recently asserted value (which is "*scheme A*" and "*Schedule 1*" in DSTC 3).

Among the slots included in DSTC 2 and DSTC 3, we used the "area," "food," and "price range" slots. A summary of the slots is shown in Table 2. We excluded "hastv" and "childrenallowed" slots because these slots have binary values ("yes" or "no") and no unseen value appeared. We also excluded the "name" slot because word vectors for several values were not obtained.

For Task 1, a model was trained using the dstc2_train dataset and evaluated using the dstc2_test dataset. The datasets include 1,612 and 1,117 sessions. Note that these two datasets use the same ontology.

For Task 2, a model was trained using the dstc3_train dataset and evaluated using the dstc3_test dataset. The dstc3_train dataset is the sum of the dstc2_train and the dstc2_test datasets. The dstc3_test dataset includes 2,264 sessions.

4.2 System Implementation

We implemented a prototype DST system based on the proposed method using the neural network framework Chainer [14]. One-best ASR results were used as inputs to the encoding layer described in Sect. 3. 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 a GloVe model [11] distributed on the GloVe website[1]. The word vectors were processed by the RNN as described in Sect. 3. The cell size of the bi-directional LSTM and NNs ($\mathrm{NN}_{\mathrm{usr}}, \mathrm{NN}_{\mathrm{sys}}$, $\mathrm{NN}_{\mathrm{pre}}, \mathrm{NN}_{\mathrm{bias}}$) was 32, and the cells of the LSTM were initialized to 0 before training.

For training the model, we used Adam with a learning rate of 0.001, gradient clipping of 1.0, mini batch size of 32, 50% dropout for the bi-directional LSTM. We used word dropout by randomly replacing a word vector with a zero vector for attention weight calculation. We trained each model by using word dropout ratios from 0% to 50% in 10 increments. Each model was trained with 200 epochs and the best parameter selected based on the accuracy for the development dataset, i.e., the dstc2_dev dataset.

For evaluation, we used a scaling parameter β for the bias b to prevent false negatives of unseen values. We replaced b with $\tilde{b} = b/\beta$ in Eq. (6). We selected the scaling parameter for each model from $[1, 2, 4, 8, 16]$ based on the performance in the DSTC 3 test dataset.

DST based on the proposed method is called as "*Cos with sentinel*" in the following parts.

4.3 Comparative Methods

We implemented three comparative methods: *"Cos without sentinel," "NN,"* and *"RNN."* Cos without sentinel used the cosine similarity-based decoder but did not use the sentinel weights to integrate the similarities. The similarities were updated by the rule $s_{n,k} = 1 - (1 - s_{n-1,k})(1 - s^{\mathrm{u}}_{n,k})$ described in [15]. NN used an NN for similarity calculation $s^{\mathrm{u}}_k = \mathrm{NN}_{\mathrm{dec}}(\mathbf{c} \oplus \mathbf{v}_k)$. RNN used an LSTM for turn-level recurrent connection in the decoding layer as $s_{n,k} = \mathrm{LSTM}_{\mathrm{dec}}\left(\mathbf{c} \oplus \mathbf{r}_k \oplus \mathbf{v}_k, s_{n-1,k}\right)$. Note that the scaling parameter was also used for all methods.

We also show two additional results of the DSTC 2 and DSTC 3 participants: "*focus baseline*" which uses rules, and "*RNN with rule*" which is the DST proposed in [4, 5]. Note that RNN with rule showed the best accuracy in DSTC 3 [2]. Results for the methods were extracted from the dstc2_results and dstc3_results distributed on the website[2].

4.4 Results

Table 3 shows the accuracy of DST in area, food and price slots in Task 1 and Task 2. We see from this table that the proposed method "Cos with Sentinel" performs better than RNN, NN, and Cos without sentinel models except for the price slot in Task 1

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

[2] http://camdial.org/ mh521/dstc

Table 3 DST accuracy for Task 1 and Task 2

	Task1			Task2			Tasks 1&2
	area	food	price	area	food	price	average
Focus baseline	90.8	83.7	92.9	81.1	90.5	88.4	87.9
RNN with rule	**92.4**	**85.6**	**93.0**	**88.5**	**91.0**	**93.2**	**90.6**
RNN	64.2	69.8	85.1	63.1	67.8	**82.9**	72.2
NN	64.5	79.7	**88.4**	57.3	68.5	79.7	73.0
Cos without sentinel	56.1	52.3	65.5	57.5	38.7	61.1	55.2
Cos with sentinel	**84.7**	**84.4**	83.7	**80.6**	**79.6**	66.9	**80.0**

Table 4 DST accuracy for unseen values

	Task2				Task2 using other unseen values			
	area	food	price	average	inexpensive	affordable	costly	pricey
Focus baseline	67.8	**88.1**	87.6	81.2	-	-	-	-
RNN with rule	**85.3**	82.3	**92.3**	**86.6**	-	-	-	-
RNN	21.4	33.1	31.1	28.5	46.2	34.5	26.9	71.7
NN	13.3	31.3	54.6	33.1	0.0	22.1	19.6	0.0
Cos without sentinel	**73.2**	31.1	**73.0**	59.1	81.9	78.5	**89.0**	**89.0**
Cos with sentinel	71.5	**59.5**	52.7	**61.2**	**83.2**	**79.3**	83.8	84.2

and Task 2. Cos without sentinel performs bad compared with other models. This is because the model failed to merge previous turns and system responses.

Table 4 shows the accuracy only for unseen values. The DST methods using the cosine similarity-based decoder (Cos without sentinel and Cos with sentinel) achieved better accuracy than those using the neural network-based decoder (NN with sentinel and RNN). This shows that the cosine similarity-based attention works for estimating values unseen in training while the neural network-based methods have difficulty in generalizing to unseen values.

The performance of the DSTC 3 participants are higher than that of the proposed method. RNN with rule performs good in particular for unseen values in the slots except the food slot in Task 2 at a cost of hand-crafting rules. The proposed method achieved a good performance without any rules.

4.5 Discussion

To understand the strengths and weaknesses of the proposed method, we analyzed performance on individual slots and values. Among the three slots, accuracy of the price range slot was 52.7%, which is lower than that of the area slot (71.5%) and the food slot (59.5%) as shown in Table 4. This result might be the effect of an incorrect word vector of the unseen value "*free*." In fact, the GloVe model gives the 5-top-most similar words for "free" as "Free," "download," "downloads," "downloadable," and "online."

To further validate the proposed method for the price range slot, we evaluated DST performance using other price-related words as unseen values: "*inexpensive*," "*affordable*," "*costly*," and "*pricey*." We replaced "free" with one of the words in the dataset and evaluated the results using the replaced dataset. The right side of Table 4 shows evaluation results for the replaced values. The replacement indicates significant improvement of the performance. We see from this that good word vectors are essential to the proposed DST method.

One of the weaknesses is detection of values that include the same word (e.g., *chinese* and *chinese takeaway*). The cosine similarity-based decoders (both Cos without sentinel and Cos with sentinel) are confused between two values and its accuracy of chinese takeaway was 0. To handle such values, we will take account of the number of words composing values as future work.

5 Conclusion

We proposed a fully data-driven approach to DST using an RNN model with an attention mechanism. The approach integrates a decoder that detects an unseen value from a user's utterance using cosine similarity between a context vector of the user's utterance and that of the unseen value, and a sentinel mixture architecture that merges an estimated dialog state of the previous turn and that of the current turn. We evaluated a DST model based on the proposed approach using the DSTC 2 and DSTC 3 datasets. For unseen values, the results showed that the use of the cosine similarity-based decoder led to a 26.0-point improvement from conventional NN-based DST, and the integration of the cosine similarity-based decoder and sentinel mechanism led to a further 2.1-point improvement. For all values, the accuracy of the conventional cosine similarity-based decoder was 17.8-point lower than that of conventional NN-based DST, but the proposed integration achieved 7.0-point improvement from the NN-based DST.

Future work includes improving accuracy for both seen and unseen values, extending the proposed approach to handle unseen slots, and handling values having no word vectors such as proper nouns.

References

[1] Henderson M., Thomson B., and Williams J. The second dialog state tracking challenge. In *Proc. of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 263, 2014.

[2] Henderson M., Thomson B., and Williams J. The third dialog state tracking challenge. In *Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE, 2014.

[3] Henderson M., Thomson B., and Young S. Deep neural network approach for the dialog state tracking challenge. In *Proc. of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 467–471, 2013.

[4] Henderson M., Thomson B., and Young S. Word-based dialog state tracking with recurrent neural networks. In *Proc. of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, 2014.

[5] Henderson M., Thomson B., and Young S. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Spoken Language Technology Workshop (SLT)*, pages 360–365. IEEE, 2014.

[6] Hori T., Wang H., Hori C., Watanabe S., Harsham B., Roux J.L., Hershey J.R., Koji Y., Jing Y., Zhu Z., et al. Dialog state tracking with attention-based sequence-to-sequence learning. In *Spoken Language Technology Workshop (SLT)*, pages 552–558. IEEE, 2016.

[7] Jang Y., Ham J., Lee B.J., Chang Y., and Kim K.E. Neural dialog state tracker for large ontologies by attention mechanism. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 531–537, Dec 2016.

[8] Kim S., DHaro L.F, Banchs R.E., Williams J.D, Henderson M., and Yoshino K. The fifth dialog state tracking challenge. In *Proc. of the 2016 IEEE Workshop on Spoken Language Technology (SLT)*, 2016.

[9] Lee S. Structured discriminative model for dialog state tracking. In *Proc. of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialog (SIGDIAL)*, pages 442–451, 2013.

[10] Merity S., Xiong C., Bradbury J., and Socher R. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016.

[11] Pennington J., Socher R., and Manning C.D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[12] Shi H., Ushio T., Endo M., Yamagami K., and Horii N. Convolutional neural networks for multi-topic dialog state tracking. In *Dialogues with Social Robots*, pages 451–463. Springer, 2017.

[13] Thomson B. and Young S. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

[14] Tokui S., Oono K., Hido S., and Clayton J. Chainer: a next-generation open source framework for deep learning. In *Proc. of workshop on machine learning systems in the 29th annual conference on neural information processing systems (NIPS)*, 2015.

[15] Wang Z. and Lemon O. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proc. of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialog (SIGDIAL)*, pages 423–432, 2013.